RAJIV GANDHI INSTITUTE OF TECHNOLOGY
GOVERNMENT ENGINEERING COLLEGE
KOTTAYAM - 686 501

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

CSL-333 DBMS LAB RECORD

Submitted by
ASWATHY S
(REG NO: KTE20CS018)

Fifth Semester B.Tech in CSE

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
THIRUVANANTHAPURAM

January 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RAJIV GANDHI INSTITUTE OF TECHNOLOGY
GOVERNMENT ENGINEERING COLLEGE
KOTTAYAM - 686 501

CERTIFICATE

This is to certify that this is a bonafide report of ***CSL 333 DBMS LAB*** *done by* ***Aswathy S (KTE20CS018)*** *towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer Science and Engineering of APJ Abdul Kalam Technological University during the year 2022-2023.*

Prof. Anil Kumar S.
Assistant Professor
Lab in Charge

Internal Examiner

External Examiner

Contents

DBMS Preliminaries

What is Database?

A Database is a collection of interrelated data stored together with controlled redundancy to serve one or more applications in an optimal way... The data are stored in such a way that they are independent of the programs used by the people for accessing the data. The approach used in adding the new data, modifying and retrieving the existing data from the database is common and controlled one.

It is also defined as a collection of logically related data stored together that is designed to meet information, requirements of an organisation. The example of a database is a Telephone Directory that contains the names, address and Phone numbers of the people stored in the Computer Storage.

Database are organised by Fields, Records and Files. These are described briefly as follows.

1. Fields: - It is the smallest unit of the data that has meaning to its users and it also called data item or data element. Name, Address, Phone Numbers are examples of Fields. These are represented in the database by a value.

2. Records: - A Record is a collection of logically related fields and each fields is processing a fixed number of bytes and is of fixed data type. The Complete information about a particular Phone Number in the data base represents a record. Records are of two types, Fixed Length Records and Variable Length records.

3. Files: - A File is a collection of related record. Generally all records in a file are of the same size and same record type, but it is not always true. The records in a file may of Fixed Length or Variable Length. Depending upon the size of the records contained in a file. The Telephone Directory containing records about the different Tele phone holders is an example of file.

Components of a Database

1. Data Item:- It is defined as a distinct piece of information and is explained in the previous section.

2. Relationships:- It represents a correspondence between various data items.

3. Constraints:- These are predicates that define correct data base states.

4. Schema:- It describes the organization of data and relationships within the database. The schema consists of definitions of various types of records in the database, the data items they contain and the set into which they are grouped. The storage structure of the database is described by the storage schema. The Conceptual Schema defines the stored data structure; the external schema defines a view of the data base for a particular user.

What is DBMS?

DBMS is a program or group of programs that work in conjunction with the operating system to create process, store, retrieve, control and manage the data. It acts as an interface between application program and the data stored in the database. Alternatively it can be defined as a computerised record keeping system that stores information and allows the users to add, delete modify, retrieve and update that information. The DBMS performs the following five primary functions.

1. Define, create and organise a database: - The DBMS establishes the logical relationships among different data elements in a database and also defines schemas and sub schemas using DDL.

2. Input Data :- It Performs the function of entering the data into the database through an input device (like data screen or voice activated system) with the help of the user.
3. Process Data:- It performs the function of manipulation and processing of the data stored in the database using DML.
4. Maintain data integrity and security:- It allows limited access of the database to authorised users to maintain data integrity and security.
5. Query database:- It provides information to the decision makers that they need To make important decisions. This information is provided by querying the database using SQL.

ER DIAGRAM

Entity

An entity is an object or concept about which you want to store information. An Entity is represented by means of a rectangle with the name of the entity specified within the rectangle.

Attribute

An Attribute of an Entity is a property that characterizes the entity.

Types of Attributes

Single Valued vs. Multivalued Attributes

An attribute with a single value for a single entity is called Single valued Attribute. An Attribute with possibly more than one value for a single entity is called multvalued attribute. A Single Valued Attribute is represented by means of an Oval with the name of the attribute specified within the oval. A Multivalued Attribute is represented by means of an Double Oval with the name of the attribute specified within the oval.

Stored vs. Derived Attributes

An attribute that must be explicitly stored within the data's is called Stored Database. A Derived Attribute is one whose value can be determined from one or more stored or derived attributes of the entity and hence need not be stored explicitly within the database. A Derived Attribute is represented by means of a dotted Ova l with the name of the attribute specified within the oval.

Atomic vs. Composite Attribute

An attribute which cannot be further divided into other attributes is called Atomic Attribute. A Composite Attribute is one which is composed of more than attributes.the representation of a Composite attribute is shown in Fig

Key attribute

A key attribute is a set of one or more attributes of an entity which can uniquely identify that entity. It is represented in the same way as other attributes are represented, but the name of the entity is underlined.

Weak Entity

An Entity with no key of its own is called a Weak Entity. It is denoted by Double Rectangle with the name of the entity specified within the inner rectangle.

Relationships

A Relationship is an association among two or more entitiy sets. Relationships illustrate how two entities share information in the database structure. A Relationship involving n Entities is called an $n - ary$ relation. If $n = 2$, the relation is called a Binary Relation. If $n = 3$, the relations is called Ternary Relation. Binary and Ternary Relations are the most common relations in RDBMS Design. Throughout the remaining discussions, the word 'relation' is used simply to represent Binary Relationship.

Cardinality

Let R be a raleation connecting two entity sets E_1 and E_2. Cardinality specifies how many instances of an entity in E_1 relate to one instance of an entity in E_2. Based on Cardinality, R can be any one of the following.

One to One Relation:- - R associates an entity in E_1 to at most entity in E_2.

One to Many Relationship:- R associates an entity in E_1 to any number of entities in E_2.

Many to Many Relationship:- R associates an entity in E_1 to any number of entities in E_2 and vice versa.

Sub classes and Super classes

In some cases, and entity type has numerous sub-groupings of its entities that are meaningful, and need to be explicitly represented, because of their importance. For example, members of entity Employee can be grouped further into Secretary, Engineer, Manager, Technician, Salaried_Employee. The set listed is a subset of the entities that belong to the Employee entity, which means that every entity that belongs to one of the sub sets is also an Employee. Each of these sub-groupings is called a subclass and the Employee entity is called the super-class. An entity cannot only be a member of a subclass; it must also be a member of the super-class. An entity can be included as a member of a number of sub classes, for example, a Secretary may also be a salaried employee.

Specialization

The process of defining a set of subclasses of a super class. Specialization is the top-down refinement into (super) classes and subclasses. The set of sub classes is based on some distinguishing characteristic of the super class. For example, the set of sub classes for Employee, Secretary, Engineer, Technician, differentiates among employee based on job type. There may be several specializations of an entity type based on different distinguishing characteristics. Another example is the specialization, Salaried_Employee and Hourly_Employee, which distinguish employees based on their method of pay.

To represent a specialization, the subclasses that define a specialization are attached by lines to a circle that represents the specialization, and is connected to the super class. The subset symbol (half- circle) is shown on

each line connecting a subclass to a super class, indicates the direction of the super class/subclass relationship. Attributes that only apply to the sub class are attached to the rectangle representing the subclass. They are called specific attributes. A sub class can also participate in specific relationship types. See Example.

Reasons of Specialization:-

- Certain attributes may apply to some but not all entities of a super class. A subclass is defined in order to group the entities to which the attributes apply.
- The second reason for using subclasses is that some relationship types may be participated in only by entities that are members of the subclass.

Generalization

The reverse of specialization is generalization. Several classes with common features are generalized into a super class. For example, the entity types Car and Truck share common attributes License_PlateNo, VehicleID and Price, therefore they can be generalized into the super class Vehicle.

Relational Database Model

The Relational Database Model represents the database as a collection of relations where each relation is a table consisting one or more columns. Formally the table is called the relation, the columns are called the attributes and the rows are called the attributes. Each attribute A in a relation can assume any value in a set of permissible values $Dom(A)$ called the Domain of A. It is usual to designate the Domain of an attribute by means of specifying a data type from which the possible data values forming the domain can be drawn. The Number of tuples in an existing relation is called the Cardinality of the relation and the number of attributes is called the *Arity* (or Degree) of the relation. A Relation R involving n attributes $A_1, A_2, A_3, \cdots, A_n$ is denoted by $R(A_1, A_2, A_3, \cdots, A_n)$ and is known as the *Schema* of the Relation. Hence it becomes apparent that, corresponding to any relation, there exists a Relation Schema.

Relational Database Guidelines

The following four guidelines are commonly followed in the design of Relational Databases.

1. The names of the attributes in Relational Table must correspond to the entity or the relation for which the table is designed.
2. While creating Relations, it is advised to avoid those attributes which stores null values more often.
3. The Relational Tables must be designed in such a way as to avoid Data Redundancy, Insertion, Deletion and Updation anomalies
4. The Relational Tables must be designed and created in such way that, appropriate foreign key references must be made in the concerned tables so as to avoid the generation of spurious tuples when taking the join of the tables.

Throughout the manual, the following procedure is used to create a database system for a given problem.

1. Identify the Entities, Attributes, Relations and the constraints in the given problem and form an ER Diagram.
2. Convert the ER Diagram to the Relational Schema using the Standard procedure for converting ER Diagram to Relational Schema.

3. Implement each schema as a table in the database with constraints modelled properly.
4. Upload sufficient information in the tables and Form the Queries, PL/SQL blocks, Cursors, Triggers, Functions and Procedures which constitute the required functionality (ies).

Structured Query Language

Tables In relational database systems (DBS) data are represented using tables (relations). A query issued against the DBS also results in a table. A table has the following structure:

Column 1 Column 2 \cdots Column n

A table is uniquely identified by its name and consists of rows that contain the stored information, each row containing exactly one tuple (or record). A table can have one or more columns. A column is made up of a column name and a data type, and it describes an attribute of the tuples. The structure of a table, also called relation schema, thus is defined by its attributes. The type of information to be stored in a table is defined by the data types of the attributes at table creation time. SQL uses the terms table, row, and column for relation, tuple, and attribute, respectively. A table can have up to 254 columns which may have different or same data types and sets of values (domains), respectively. Possible domains are alphanumeric data (strings), numbers and date formats. Oracle offers the following basic data types:

1. char(n): Fixed-length character data (string), n characters long. The maximum size for n is 255 bytes (2000 in Oracle8). Note that a string of type char is always padded on right with blanks to full length of n. (can be memory consuming). Example: char(40)
2. varchar2(n): Variable-length character string. The maximum size for n is 2000 (4000 in Oracle8). Only the bytes used for a string require storage.
3. date: Date data type for storing date and time. The default format for a date is: DD-MMM-YY. Examples: '13-OCT-94', '07-JAN-98'
4. long: Character data up to a length of 2GB. Only one long column is allowed per table.

As long as no constraint restricts the possible values of an attribute, it may have the special value null (for unknown). This value is different from the number 0, and it is also different from the empty string ''. Further properties of tables are:

1. the order in which tuples appear in a table is not relevant (unless a query requires an explicit sorting).
2. a table has no duplicate tuples (depending on the query, however, duplicate tuples can appear in the query result).

A database schema is a set of relation schemas. The extension of a database schema at database run-time is called a database instance or database, for short.

In order to retrieve the information stored in the database, the SQL query language is used. In SQL a query has the following (simplified) form (components in brackets [ ] are optional):

```
select [distinct] <column(s)>
from <table>
[ where <condition> ]
[ order by <column(s) [asc|desc]> ]
```

Selecting Columns The columns to be selected from a table are specified after the keyword select. This operation is also called projection. For example, the query

```
select LOC, DEPTNO from DEPT;
```

lists only the number and the location for each tuple from the relation DEPT. If all columns should be selected, the asterisk symbol "" can be used to denote all attributes. The query

```
select  from EMP;
```

retrieves all tuples with all columns from the table EMP. Instead of an attribute name, the select clause may also contain arithmetic expressions involving arithmetic operators etc.

```
select ENAME, DEPTNO, SAL  1.55 from EMP;
```

Consider the query

```
select DEPTNO from EMP;
```

which retrieves the department number for each tuple. Typically, some numbers will appear more than only once in the query result, that is, duplicate result tuples are not automatically eliminated. Inserting the keyword distinct after the keyword select, however, forces the elimination of duplicates from the query result.

It is also possible to specify a sorting order in which the result tuples of a query are displayed. For this the order by clause is used and which has one or more attributes listed in the select clause as parameter. desc specifies a descending order and asc specifies an ascending order (this is also the default order). For example, the query

```
select ENAME, DEPTNO, HIREDATE from EMP;
from EMP
order by DEPTNO [asc], HIREDATE desc;
```

displays the result in an ascending order by the attribute DEPTNO. If two tuples have the same attribute value for DEPTNO, the sorting criteria is a descending order by the attribute values of HIREDATE.

Selection of Tuples Up to now we have only focused on selecting (some) attributes of all tuples from a table. If one is interested in tuples that satisfy certain conditions, the where clause is used. In a where clause simple conditions based on comparison operators can be combined using the logical connectives and, or, and not to form complex conditions. Conditions may also include pattern matching operations and even subqueries

For all data types, the comparison operators $=, != or <>, <, >, <=, =>$ are allowed in the conditions of a where clause. Further comparison operators are:

1. Set Conditions: <column> [not] in (<list of values>) Example: select from DEPT where DEPTNO in (20,30);

2. Null value: <column> is [not] null, i.e., for a tuple to be selected there must (not) exist a defined value for this column. Example:

   select from EMP where MGR is not null;

3. Domain conditions: <column> [not] between <lower bound> and <upper bound> Example:

   ```
   select EMPNO, ENAME, SAL from EMP
   where SAL between 1500 and 2500;
   ```

   ```
   select ENAME from EMP
   where HIREDATE between '02-APR-81' and '08-SEP-81';
   ```

String Operations In order to compare an attribute with a string, it is required to surround the string by apostrophes. A powerful operator for pattern matching is the like operator. Together with this operator, two special characters are used: the percent sign % (also called wild card), and the underline , also called position marker. For example, if one is interested in all tuples of the table DEPT that contain two C in the name of the department, the condition would be where DNAME like '%C%C%'. The percent sign means that any (sub)string is allowed there, even the empty string. Further string operations are:

1. upper(<string>) takes a string and converts any letters in it to uppercase,
2. lower(<string>) converts any letter to lowercase,
3. initcap(<string>) converts the initial letter of every word in <string> to uppercase.
4. length(<string>) returns the length of the string.
5. substr(<string>, n [, m]) clips out a m character piece of <string>, starting at position n. If m is not specified, the end of the string is assumed.

1.2.4 Aggregate Functions Aggregate functions are statistical functions such as count, min, max etc. They are used to compute a single value from a set of attribute values of a column:

Function name	Operation
AVG	calculates the average of non-NULL values in a set
COUNT	returns the number of rows in a group, including rows with NULL values
MAX	returns the highest value (maximum) in a set of non-NULL values.
MIN	returns the lowest value (minimum) in a set of non-NULL values.
SUM	returns the summation of all non-NULL values a set

Creating Tables The SQL command for creating an empty table has the following form:

```
create table <table> (
<column 1> <data type> [not null] [unique] [<column constraint>],
. . . . . .
<column n> <data type> [not null] [unique] [<column constraint>],
[<table constraint(s)>]
);
```

For each column, a name and a data type must be specified and the column name must be unique within the table definition. Column definitions are separated by comma. There is no difference between names in lower case letters and names in upper case letters. In fact, the only place where upper and lower case letters matter are strings comparisons. A not null constraint is directly specified after the data type of the column and the constraint requires defined attribute values for that column, different from null. The keyword unique specifies that no two tuples can have the same attribute value for this column. Unless the condition not null is also specified for this column, the attribute value null is allowed and two tuples having the attribute value null for this column do not violate the constraint.

Specifying Constraints The definition of a table may include the specification of integrity constraints. Basically two types of constraints are provided: column constraints are associated with a single column whereas table constraints are typically associated with more than one column.

The specification of a (simple) constraint has the following form:

```
[constraint <name>] primary key | unique | not null
```

A constraint can be named. It is advisable to name a constraint in order to get more meaningful information when this constraint is violated due to, e.g., an insertion of a tuple that violates the constraint.

The two most simple types of constraints have already been discussed: not null and unique. Probably the most important type of integrity constraints in a database are primary key constraints. A primary key constraint enables a unique identification of each tuple in a table. Based on a primary key, the database system ensures that no duplicates appear in a table. For example, for our EMP table, the specification

```
create table EMP (
EMPNO number(4) constraint pk emp primary key,
. . . );
```

defines the attribute EMPNO as the primary key for the table. Each value for the attribute EMPNO thus must appear only once in the table EMP. A table, of course, may only have one primary key. Note that in contrast to a unique constraint, null values are not allowed.

Example:

We want to create a table called PROJECT to store information about projects. For each project, we want to store the number and the name of the project, the employee number of the project's manager, the budget and the number of persons working on the project, and the start date and end date of the project. Furthermore, we have the following conditions:

- a project is identified by its project number,
- the name of a project must be unique,
- the manager and the budget must be defined.

The table can be created as follows.

```
create table PROJECT (
PNO number(3) constraint prj pk primary key,
PNAME varchar2(60) unique,
PMGR number(4) not null,
PERSONS number(5),
BUDGET number(8,2) not null,
PSTART date,
PEND date);
```

Integrity Constraints In this section we introduce two ypes of constraints that can be specified within the create table statement: check constraints (to restrict possible attribute values), and foreign key constraints (to specify interdependencies between relations).

Check Constraints Often columns in a table must have values that are within a certain range or that satisfy certain conditions. Check constraints allow users to restrict possible attribute values for a column to admissible ones. They can be specified as column constraints or table constraints. The syntax for a check constraint is

```
[constraint <name>] check(<condition>)
```

If a check constraint is specified as a column constraint, the condition can only refer that column. Example: The name of an employee must consist of upper case letters only; the minimum salary of an employee is 500; department numbers must range between 10 and 100:

```
create table EMP
( . . . ,
ENAME varchar2(30) constraint check name
check(ENAME = upper(ENAME) ),
SAL number(5,2) constraint check sal check(SAL >= 500),
DEPTNO number(3) constraint check deptno
check(DEPTNO between 10 and 100) );
```

Foreign Key Constraints A foreign key constraint (or referential integrity constraint) can be specified as a column constraint or as a table constraint:

```
[constraint <name>] [foreign key (<column(s)>)]
references <table>[(<column(s)>)]
[on delete cascade]
```

This constraint specifies a column or a list of columns as a foreign key of the referencing table. The referencing table is called the child-table, and the referenced table is called the parent-table. In other words, one cannot define a referential integrity constraint that refers to a table R before that table R has been created. The clause foreign key has to be used in addition to the clause references if the foreign key includes more than one column. In this case, the constraint has to be specified as a table constraint. The clause references defines which columns of the parent-table are referenced. If only the name of the parent-table is given, the list of attributes that build the primary key of that table is assumed. Example: Each employee in the table EMP must work in a department that is contained in the table DEPT:

```
create table EMP
( EMPNO number(4) constraint pk emp primary key,
. . . ,
DEPTNO number(3) constraint fk deptno references DEPT(DEPTNO) );
```

The column DEPTNO of the table EMP (child-table) builds the foreign key and references the primary key of the table DEPT (parent-table).

Insertions The most simple way to insert a tuple into a table is to use the insert statement

```
insert into <table> [(<column i, . . . , column j>)]
values (<value i, . . . , value j>);
```

For each of the listed columns, a corresponding (matching) value must be specified. Thus an insertion does not necessarily have to follow the order of the attributes as specified in the create table statement. If a column is omitted, the value null is inserted instead. If no column list is given, however, for each column as defined in the create table statement a value must be given. Examples:

```
insert into PROJECT(PNO, PNAME, PERSONS, BUDGET, PSTART)
values(313, 'DBS', 4, 150000.42, '10-OCT-94');
or
insert into PROJECT
values(313, 'DBS', 7411, null, 150000.42, '10-OCT-94', null);
```

If there are already some data in other tables, these data can be used for insertions into a new table. For this, we write a query whose result is a set of tuples to be inserted. Such an insert statement has the form

```
insert into <table> [(<column i, . . . , column j>)] <query>
```

Example: Suppose we have defined the following table:

```
create table OLDEMP (
ENO number(4) not null,
HDATE date);
```

We now can use the table EMP to insert tuples into this new relation:

```
insert into OLDEMP (ENO, HDATE)
select EMPNO, HIREDATE from EMP
where HIREDATE < '31-DEC-60';
```

Updates For modifying attribute values of (some) tuples in a table, we use the update statement:

```
update <table> set
<column i> = <expression i>, . . . , <column j> = <expression j>
[where <condition>];
```

An expression consists of either a constant (new value), an arithmetic or string operation, or an SQL query. Note that the new value to assign to <column i> must a the matching data type. An update statement without a where clause results in changing respective attributes of all tuples in the specified table. Typically, however, only a (small) portion of the table requires an update. Examples:

1. The employee JONES is transfered to the department 20 as a manager and his salary is increased by 1000:

   ```
   update EMP set
   JOB = 'MANAGER', DEPTNO = 20, SAL = SAL +1000
   where ENAME = 'JONES';
   ```

2. All employees working in the departments 10 and 30 get a 15update EMP set

   ```
   SAL = SAL  1.15 where DEPTNO in (10,30);
   ```

Analogous to the insert statement, other tables can be used to retrieve data that are used as new values. In such a case we have a <query> instead of an <expression>. Example: All salesmen working in the department 20 get the same salary as the manager who has the lowest salary among all managers.

```
update EMP set
SAL = (select min(SAL) from EMP
where JOB = 'MANAGER')
where JOB = 'SALESMAN' and DEPTNO = 20;
```

Explanation: The query retrieves the minimum salary of all managers. This value then is assigned to all salesmen working in department 20.

Deletions All or selected tuples can be deleted from a table using the delete command:

```
delete from <table> [where <condition>];
```

If the where clause is omitted, all tuples are deleted from the table. An alternative command for deleting all tuples from a table is the truncate table <table> command. However, in this case, the deletions cannot be undone (see subsequent Section 1.4.4). Example: Delete all projects (tuples) that have been finished before the actual date (system date):

```
delete from PROJECT where PEND < sysdate;
```

sysdate is a function in SQL that returns the system date. Another important SQL function is user, which returns the name of the user logged into the current Oracle session. *Queries involving multiple tables A major feature of relational databases is to combine (join) tuples stored in different tables in order to display more meaningful and complete information. In SQL the select statement is used for this kind of queries joining relations:

```
select [distinct] [<alias ak>.]<column i>, . . . , [<alias al>.]<column j>
from <table 1> [<alias a1>], . . . , <table n> [<alias an>]
[where <condition>]
```

The specification of table aliases in the from clause is necessary to refer to columns that have the same name in different tables. For example, the column DEPTNO occurs in both EMP and DEPT. If we want to refer to either of these columns in the where or select clause, a table alias has to be specified and put in the front of the column name. Instead of a table alias also the complete relation name can be put in front of the column such as DEPT.DEPTNO, but this sometimes can lead to rather lengthy query formulations.

Joining Relations Comparisons in the where clause are used to combine rows from the tables listed in the from clause. Example: In the table EMP only the numbers of the departments are stored, not their name. For each salesman, we now want to retrieve the name as well as the number and the name of the department where he is working:

```
select ENAME, E.DEPTNO, DNAME
from EMP E, DEPT D
where E.DEPTNO = D.DEPTNO
and JOB = 'SALESMAN';
```

Explanation: E and D are table aliases for EMP and DEPT, respectively. The computation of the query result occurs in the following manner (without optimization):

1. Each row from the table EMP is combined with each row from the table DEPT (this operation is called Cartesian product). If EMP contains m rows and DEPT contains n rows, wethus get nm rows.
2. From these rows those that have the same department number are selected (where E.DEPTNO = D.DEPTNO).
3. From this result finally all rows are selected for which the condition JOB = 'SALESMAN' holds.

In this example the joining condition for the two tables is based on the equality operator "=". The columns compared by this operator are called join columns and the join operation is called an equijoin.

Any number of tables can be combined in a select statement.

Example: For each project, retrieve its name, the name of its manager, and the name of the department where the manager is working:

```
select ENAME, DNAME, PNAME
from EMP E, DEPT D, PROJECT P
where E.EMPNO = P.MGR
and D.DEPTNO = E.DEPTNO;
```

It is even possible to join a table with itself: Example: List the names of all employees together with the name of their manager:

```
select E1.ENAME, E2.ENAME
from EMP E1, EMP E2
where E1.MGR = E2.EMPNO;
```

Subqueries A query result can also be used in a condition of a where clause. In such a case the query is called a subquery and the complete select statement is called a nested query.

Example: List the name and salary of employees of the department 20 who are leading a project that started before December 31, 1990:

```
select ENAME, SAL from EMP
where EMPNO in
(select PMGR from PROJECT
where PSTART < '31-DEC-90')
and DEPTNO =20;
```

Explanation: The subquery retrieves the set of those employees who manage a project that started before December 31, 1990. If the employee working in department 20 is contained in this set (in operator), this tuple belongs to the query result set.

Grouping Often applications require grouping rows that have certain properties and then applying an aggregate function on one column for each group separately. For this, SQL provides the clause group by <group column(s)>. This clause appears after the where clause and must refer to columns of tables listed in the from clause.

```
select <column(s)>
from <table(s)>
where <condition>
group by <group column(s)>
[having <group condition(s)>];
```

Those rows retrieved by the selected clause that have the same value(s) for <group column(s)> are grouped. Aggregations specified in the select clause are then applied to each group separately. It is important that only those columns that appear in the <group column(s)> clause can be listed without an aggregate function in the select clause ! Example: For each department, we want to retrieve the minimum and maximum salary.

```
select DEPTNO, min(SAL), max(SAL)
from EMP
group by DEPTNO;
```

Rows from the table EMP are grouped such that all rows in a group have the same department number. The aggregate functions are then applied to each such group.

Rows to form a group can be restricted in the where clause. For example, if we add the condition where JOB = 'CLERK', only respective rows build a group. The query then would retrieve the minimum and maximum salary of all clerks for each department. Note that is not allowed to specify any other column than DEPTNO without an aggregate function in the select clause since this is the only column listed in the group by clause (is it also easy to see that other columns would not make any sense).

Once groups have been formed, certain groups can be eliminated based on their properties, e.g., if a group contains less than three rows. This type of condition is specified using the having clause. As for the select clause also in a having clause only <group column(s)> and aggregations can be used.

Example: Retrieve the minimum and maximum salary of clerks for each department having more than three clerks.

```
select DEPTNO, min(SAL), max(SAL)
from EMP
where JOB = 'CLERK'
group by DEPTNO
having count() > 3;
```

Note that it is even possible to specify a subquery in a having clause. In the above query, for example, instead of the constant 3, a subquery can be specified. A query containing a group by clause is processed in the following way:

1. Select all rows that satisfy the condition specified in the where clause.
2. From these rows form groups according to the group by clause.
3. Discard all groups that do not satisfy the condition in the having clause.
4. Apply aggregate functions to each group.
5. Retrieve values for the columns and aggregations listed in the select clause.

Modifying Table- and Column Definitions It is possible to modify the structure of a table (the relation schema) even if rows have already been inserted into this table. A column can be added using the alter table command

```
alter table <table>
add(<column> <data type> [default <value>] [<column constraint>]);
```

If more than only one column should be added at one time, respective add clauses need to be separated by colons. A table constraint can be added to a table using

```
alter table <table> add (<table constraint>);
```

Note that a column constraint is a table constraint, too. not null and primary key constraints can only be added to a table if none of the specified columns contains a null value. Table definitions can be modified in an analogous way. This is useful, e.g., when the size of strings that can be stored needs to be increased. The syntax of the command for modifying a column is

```
alter table <table>
modify(<column> [<data type>] [default <value>] [<column constraint>]);
```

It is possible to rename tables or columns using the alter table command. A table and its rows can be deleted by issuing the command

```
drop table <table> [cascade
constraints];.
```

Views In Oracle the SQL command to create a view (virtual table) has the form

```
create [or replace] view <view-name> [(<column(s)>)] as
<select-statement> [with check option [constraint <name>]];
```

The optional clause or replace re-creates the view if it already exists. <column(s)> names the columns of the view. If <column(s)> is not specified in the view definition, the columns of the view get the same names as the attributes listed in the select statement (if possible). Example: The following view contains the name, job title and the annual salary of employees working in the department 20:

```
create view DEPT20 as
select ENAME, JOB, SAL12 ANNUAL SALARY from EMP
where DEPTNO = 20;
```

In the select statement the column alias ANNUAL SALARY is specified for the expression SAL12 and this alias is taken by the view. An alternative formulation of the above view definition is

```
create view DEPT20 (ENAME, JOB, ANNUAL SALARY) as
select ENAME, JOB, SAL  12 from EMP
where DEPTNO = 20;
```

A view can be used in the same way as a table, that is, rows can be retrieved from a view (also respective rows are not physically stored, but derived on basis of the select statement in the view definition), or rows can even be modified. A view is evaluated again each time it is accessed. In Oracle SQL no insert, update, or delete modifications on views are allowed that use one of the following constructs in the view definition:

1. Joins
2. Aggregate function such as sum, min, max etc.
3. set-valued subqueries (in, any, all) or test for existence (exists)
4. group by clause or distinct clause

In combination with the clause with check option any update or insertion of a row into the view is rejected if the new/modified row does not meet the view definition, i.e., these rows would not be selected based on the select statement. A with check option can be named using the constraint clause. A view can be deleted using the command

```
delete <view-name>.
```

PL/SQL

Introduction The development of database applications typically requires language constructs similar to those that can be found in programming languages such as C, C++, or Pascal. These constructs are necessary in order to implement complex data structures and algorithms. A major restriction of the database language SQL, however, is that many tasks cannot be accomplished by using only the provided language elements.

PL/SQL (Procedural Language/SQL) is a procedural extension of Oracle-SQL that offers language constructs similar to those in imperative programming languages. PL/SQL allows users and designers to develop complex database applications that require the usage of control structures and procedural elements such as procedures, functions, and modules.

The basic construct in PL/SQL is a block. Blocks allow designers to combine logically related (SQL-) statements into units. In a block, constants and variables can be declared, and variables can be used to store query results. Statements in a PL/SQL block include SQL statements, control structures (loops), condition statements (if-then-else), exception handling, and calls of other PL/SQL blocks. PL/SQL blocks that specify procedures and functions can be grouped into packages. A package is similar to a module and has an interface and an implementation part. Oracle offers several predefined packages, for example, input/output routines, file handling, job scheduling etc.

Another important feature of PL/SQL is that it offers a mechanism to process query results in a tuple-oriented way, that is, one tuple at a time. For this, cursors are used. A cursor basically is a pointer to a query result and is used to read attribute values of selected tuples into variables. A cursor typically is used in combination with a loop construct such that each tuple read by the cursor can be processed individually. In summary, the major goals of PL/SQL are to

1. increase the expressiveness of SQL,
2. process query results in a tuple-oriented way,
3. optimize combined SQL statements,
4. develop modular database application programs,

5. reuse program code, and

6. reduce the cost for maintaining and changing applications.

PL/SQL is a block-structured language. Each block builds a (named) program unit, and blocks can be nested. Blocks that build a procedure, a function, or a package must be named. A PL/SQL block has an optional declare section, a part containing PL/SQL statements, and an optional exception-handling part. Thus the structure of a PL/SQL looks as follows.

```
[<Block header>]
[declare
<Constants>
<Variables>
<Cursors>
<User defined exceptions>]
begin
<PL/SQL statements>
[exception
<Exception handling>]
end;
```

The block header specifies whether the PL/SQL block is a procedure, a function, or a package. If no header is specified, the block is said to be an anonymous PL/SQL block. Each PL/SQL block again builds a PL/SQL statement. Thus blocks can be nested like blocks in conventional programming languages. The scope of declared variables (i.e., the part of the program in which one can refer to the variable) is analogous to the scope of variables in programming languages such as C or Pascal.

Declarations Constants, variables, cursors, and exceptions used in a PL/SQL block must be declared in the declare section of that block. Variables and constants can be declared as follows: <variable name> [constant] <data type> [not null] [:= <expression>]; Valid data types are SQL data types (see Section 1.1) and the data type boolean. Boolean data may only be true, false, or null. The not null clause requires that the declared variable must always have a value different from null. <expression> is used to initialize a variable. If no expression is specified, the value null is assigned to the variable. The clause constant states that once a value has been assigned to the variable, the value cannot be changed (thus the variable becomes a constant). Example:

```
declare
hire date date; /* implicit initialization with null */
job title varchar2(80) := 'Salesman';
emp found boolean; /* implicit initialization with null */
salary incr constant number(3,2) := 1.5; /* constant */
. . .
begin . . . end;
```

Instead of specifying a data type, one can also refer to the data type of a table column (so-called anchored declaration). For example, EMP.EmpnoEmpno in the relation EMP. Instead of a single variable, a record can be declared that can store a complete tuple from a given table (or query result). For example, the data type DEPTspecifies a record suitable to store all attribute values of a complete row from the table DEPT. Such records are typically used in combination with a cursor. A field in a record can be accessed using

```
<record name>.<column name>, for example, DEPT.Deptno.
```

A cursor declaration specifies a set of tuples (as a query result) such that the tuples can be processed in a tuple-oriented way (i.e., one tuple at a time) using the fetch statement. A cursor declaration has the form

```
cursor <cursor name> [(<list of parameters>)] is <select statement>;
```

The cursor name is an undeclared identifier, not the name of any PL/SQL variable. A parameter has the form <parameter name> <parameter type>. Possible parameter types are char, varchar2, number, date and boolean as well as corresponding subtypes such as integer. Parameters are used to assign values to the variables that are given in the select statement. Example: We want to retrieve the following attribute values from the table EMP in a tupleoriented way: the job title and name of those employees who have been hired after a given date, and who have a manager working in a given department.

```
cursor employee cur (start date date, dno number) is
select JOB, ENAME from EMP E where HIREDATE > start date
and exists (select  from EMP
where E.MGR = EMPNO and DEPTNO = dno);
```

If (some) tuples selected by the cursor will be modified in the PL/SQL block, the clause for update[(<column(s)>)] has to be added at the end of the cursor declaration. In this case selected tuples are locked and cannot be accessed by other users until a commit has been issued. Before a declared cursor can be used in PL/SQL statements, the cursor must be opened, and after processing the selected tuples the cursor must be closed.

Language Elements In addition to the declaration of variables, constants, and cursors, PL/SQL offers various language constructs such as variable assignments, control structures (loops, if-then-else), procedure and function calls, etc. However, PL/SQL does not allow commands of the SQL data definition language such as the create table statement. For this, PL/SQL provides special packages. Furthermore, PL/SQL uses a modified select statement that requires each selected tuple to be assigned to a record (or a list of variables). There are several alternatives in PL/SQL to a assign a value to a variable. The most simple way to assign a value to a variable is

```
declare
counter integer := 0;
. . .
begin
counter := counter + 1;
```

Values to assign to a variable can also be retrieved from the database using a select statement

```
select <column(s)> into <matching list of variables>
from <table(s)> where <condition>;
```

It is important to ensure that the select statement retrieves at most one tuple ! Otherwise it is not possible to assign the attribute values to the specified list of variables and a runtime error occurs. If the select statement retrieves more than one tuple, a cursor must be used instead. Furthermore, the data types of the specified variables must match those of the retrieved attribute values. For most data types, PL/SQL performs an automatic type conversion (e.g., from integer to real). Instead of a list of single variables, a record can be given after the keyword into. Also in this case, the select statement must retrieve at most one tuple ! declare

```
employee rec EMP%ROWTYPE;
max sal EMP.SAL%TYPE;
begin
select EMPNO, ENAME, JOB, MGR, SAL, COMM, HIREDATE, DEPTNO
into employee rec
from EMP where EMPNO = 5698;
select max(SAL) into max sal from EMP;
. . .
end;
```

PL/SQL provides while-loops, two types of for-loops, and continuous loops. Latter ones are used in combination with cursors. All types of loops are used to execute a sequence of statements multiple times. The specification of loops occurs in the same way as known from imperative programming languages such as C or Pascal. A while-loop has the pattern

```
[<< <label name> >>]
while <condition> loop
<sequence of statements>;
end loop [<label name>] ;
```

A loop can be named. Naming a loop is useful whenever loops are nested and inner loops are completed unconditionally using the exit <label name>; statement. Whereas the number of iterations through a while loop is unknown until the loop completes, the number of iterations through the for loop can be specified using two integers.

```
[<< <label name> >>]
for <index> in [reverse] <lower bound>..<upper bound> loop
<sequence of statements>
end loop [<label name>] ;
```

The loop counter <index> is declared implicitly. The scope of the loop counter is only the for loop. It overrides the scope of any variable having the same name outside the loop. Inside the for loop, <index> can be referenced like a constant. <index> may appear in expressions, but one cannot assign a value to <index>. Using the keyword reverse causes the iteration to proceed downwards from the higher bound to the lower bound. Processing Cursors: Before a cursor can be used, it must be opened using the open statement

```
open <cursor name> [(<list of parameters>)] ;
```

The associated select statement then is processed and the cursor references the first selected tuple. Selected tuples then can be processed one tuple at a time using the fetch command

```
fetch <cursor name> into <list of variables>;
```

The fetch command assigns the selected attribute values of the current tuple to the list of variables. After the fetch command, the cursor advances to the next tuple in the result set. Note that the variables in the list must have the same data types as the selected values. After all tuples have been processed, the close command is used to disable the cursor.

```
close <cursor name>;
```

The example below illustrates how a cursor is used together with a continuous loop:

```
declare
cursor emp cur is select  from EMP;
emp rec EMP%ROWTYPE;
emp sal EMP.SAL%TYPE;
begin
open emp cur;
loop
fetch emp cur into emp rec;
exit when emp cur%NOTFOUND;
emp sal := emp rec.sal;
<sequence of statements>
end loop;
```

```
close emp cur;
. . .
end;
```

Each loop can be completed unconditionally using the exit clause:

```
exit [<block label>] [when <condition>]
```

Using exit without a block label causes the completion of the loop that contains the exit statement. A condition can be a simple comparison of values. In most cases, however, the condition refers to a cursor. In the example above, %NOTFOUND is a predicate that evaluates to false if the most recent fetch command has read a tuple. The value of <cursor name>before the first tuple is fetched. The predicate evaluates to true if the most recent fetch failed to return a tuple, and false otherwise. %FOUND is the logical opposite of

Cursor for loops can be used to simplify the usage of a cursor:

```
[<< <label name> >>]
for <record name> in <cursor name>[(<list of parameters>)] loop
<sequence of statements>
end loop [<label name>];
```

A record suitable to store a tuple fetched by the cursor is implicitly declared. Furthermore, this loop implicitly performs a fetch at each iteration as well as an open before the loop is entered and a close after the loop is left. If at an iteration no tuple has been fetched, the loop is automatically terminated without an exit. It is even possible to specify a query instead of <cursor name> in a for loop:

```
for <record name> in (<select statement>) loop
<sequence of statements>
end loop;
```

That is, a cursor needs not be specified before the loop is entered, but is defined in the select statement.

```
Example:
for sal rec in (select SAL + COMM total from EMP) loop
. . . ;
end loop;
```

total is an alias for the expression computed in the select statement. Thus, at each iteration only one tuple is fetched. The record sal rec, which is implicitly defined, then contains only one entry which can be accessed using sal rec.total. Aliases, of course, are not necessary if only attributes are selected, that is, if the select statement contains no arithmetic operators or aggregate functions. For conditional control, PL/SQL offers if-then-else constructs of the pattern

```
if <condition> then <sequence of statements>
[elsif] <condition> then <sequence of statements>
. . .
[else] <sequence of statements> end if;
```

Starting with the first condition, if a condition yields true, its corresponding sequence of statements is executed, otherwise control is passed to the next condition. Thus the behavior of this type of PL/SQL statement is analogous to if-then-else statements in imperative programming languages. Except data definition language commands such as create table, all types of SQL statements can be used in PL/SQL blocks, in particular delete, insert, update, and commit. Note that in PL/SQL only select statements of the type select <column(s)> into are allowed, i.e., selected attribute values can only be assigned to variables (unless the select statement is used in a subquery). The usage of select statements as in SQL leads to a syntax error. If update or delete statements are used in combination

with a cursor, these commands can be restricted to currently fetched tuple. In these cases the clause where current of<cursor name> is added as shown in the following example. Example: The following PL/SQL block performs the following modifications: All employees having 'KING' as their manager get a 5% salary increase.

```
declare
manager EMP.MGR%TYPE;
cursor emp cur (mgr no number) is
select SAL from EMP
where MGR = mgr no
for update of SAL;
begin
select EMPNO into manager from EMP
where ENAME = 'KING';
for emp rec in emp cur(manager) loop
update EMP set SAL = emp rec.sal  1.05
where current of emp cur;
end loop;
commit;
end;
```

Triggers

The different types of integrity constraints discussed so far provide a declarative mechanism to associate "simple" conditions with a table such as a primary key, foreign keys or domain constraints. Complex integrity constraints that refer to several tables and attributes (as they are known as assertions in the SQL standard) cannot be specified within table definitions. Triggers, in contrast, provide a procedural technique to specify and maintain integrity constraints. Triggers even allow users to specify more complex integrity constraints since a trigger essentially is a PL/SQL procedure. Such a procedure is associated with a table and is automatically called by the database system whenever a certain modification (event) occurs on that table. Modifications on a table may include insert, update, and delete operations (Oracle 7).

A trigger definition consists of the following (optional) components:

1. trigger name

   ```
   create [or replace] trigger <trigger name>
   ```

2. trigger time point

   ```
   before | after
   ```

3. triggering event(s)

   ```
   insert or update [of <column(s)>] or delete on <table>
   ```

4. trigger type (optional)

   ```
   for each row
   ```

5. trigger restriction (only for for each row triggers !)

   ```
   when (<condition>)
   ```

6. trigger body <PL/SQL block>

Example Trigger Suppose we have to maintain the following integrity constraint: "The salary of an employee different from the president cannot be decreased and must also not be increased more than 10%. Furthermore, depending on the job title, each salary must lie within a certain salary range.

We assume a table SALGRADE that stores the minimum (MINSAL) and maximum (MAXSAL) salary for each job title (JOB). Since the above condition can be checked for each employee individually, we define the following row trigger: trig1.sql

```
create or replace trigger check salary EMP
after insert or update of SAL, JOB on EMP
for each row
when (new.JOB != 'PRESIDENT') - - trigger restriction
declare
minsal, maxsal SALGRADE.MAXSAL%TYPE;
begin
- - retrieve minimum and maximum salary for JOB
select MINSAL, MAXSAL into minsal, maxsal from SALGRADE
where JOB = :new.JOB;
- - If the new salary has been decreased or does not lie within the salary range,
- - raise an exception
if (:new.SAL < minsal or :new.SAL > maxsal) then
raise application error(-20225, 'Salary range exceeded');
elsif (:new.SAL < :old.SAL) then
raise application error(-20230, 'Salary has been decreased');
elsif (:new.SAL > 1.1  :old.SAL) then
raise application error(-20235, 'More than 10% salary increase');
end if;
end;
```

ORACLE 10g

Oracle Application Server 10g provides full support for the Java 2 Platform Enterprise Edition (J2EE), XML, and emerging Web services standards. With Oracle Application Server 10g, you can simplify information access for your customers and trading partners by delivering enterprise portals, which can be customized and accessed from a network browser or wireless devices. It allows you to redefine your business processes, and integrate your applications and data sources with those from your customers or partners. You can deliver tailored customer experiences via real-time personalization, and assess and correlate Web site traffic patterns using Oracle Application Server 10g integrated business intelligence services.The Oracle database is a broad and powerful product. To give some structure to the broad spectrum of the Oracle database, we've organized the features into the following sections:

- Database Application Development Features The main use of the Oracle database system is to store and retrieve data for applications. This section is divided into two categories database programming and database extensibility options.
- Database Connection Features The connection between the client and the database server is a key component of the overall architecture of a computing system. The database connection is responsible for supporting all communications between an application and the data it uses.
- Distributed Database Features One of the strongest features of the Oracle database is its ability to scale up to handle extremely large volumes of data and users. Oracle scales not only by running on more and

more powerful platforms, but also by running in a distributed configuration. Oracle databases on separate platforms can be combined to act as a single logical distributed database.

- Data Movement Features Moving data from one Oracle database to another is often a requirement when using distributed databases, or when a user wants to implement multiple copies of the same database in multiple locations to reduce network traffic or increase data availability.
- Performance Features Oracle includes several features specifically designed to boost performance in certain situations.
- Database Management Features Oracle includes many features that make the database easier to manage. We've divided the discussion in this section into four categories: Oracle Enterprise Manager, add-on packs, backup and recovery, and database availability.

Experiment 1

Faculty Database
Date:-30/09/2022

Question:- Create the following two tables:
College consisting of college-code, college-name, address
Faculty consisting of fields College-code, faculty-code, faculty-name, qualification, experience (in no. of years), department, address.
The field college-code is foreign key. Generate Queries for the following.
(a) List all faulty members of a specified college whose experience is greater than or equal to 10 years.
(b) List all Faculty Members of a specified college who have at least 10 years of experience but not having M.Tech Degree.
(c) List out the Faculty of a specified college department wise in non decreasing order of their seniority.
(d) List out the Names of the Colleges having more than a specified number of faculty members.
(e) List out the names of the colleges having the least number of faculties and the largest number of faculty.

Answer:-
The SQL script for solving the given question is the following.

```
CREATE DATABASE campus;
USE campus;

CREATE TABLE College(
        collegecode VARCHAR(15) PRIMARY KEY,
        collegename VARCHAR(15),
        address VARCHAR(20)
);

CREATE TABLE Faculty(
        collegecode VARCHAR(15),
        facultycode VARCHAR(10) PRIMARY KEY,
        facultyname VARCHAR(20),
        qualification VARCHAR(20),
        experience INT,
        department VARCHAR(20),
        address VARCHAR(20),
        FOREIGN KEY (collegecode) REFERENCES College(collegecode)
);

INSERT INTO College VALUES("KTE","RIT,Kottayam","Vellor P.O, Pampady");
INSERT INTO College VALUES("GECT","GEC,Thrissur","Thiruvanathapuram");
INSERT INTO College VALUES("WYD","GEC,Wayanad","Wayanad");

INSERT INTO Faculty VALUES("KTE","9087","Priya","MTECH",15,"CSE","ABCD ,
↪  686539");
INSERT INTO Faculty VALUES("KTE","9987","Ajith","BTECH",6,"CSE","SDFT ,
↪  634429");
```

```
INSERT INTO Faculty VALUES("WYD","8465","Nitya","MTECH",20,"MCA","DFGJW,
↪  634328");
INSERT INTO Faculty VALUES("KTE","9004","Swathi","BTECH",11,"EEE","HJDJD,
↪  623990");
INSERT INTO Faculty VALUES("KTE","8924","Ajitha","MTECH",17,"EEE","REIFT,
↪  633280");
INSERT INTO Faculty VALUES("GECT","8745","Serah","MTECH",28,"MCA","DFGJWT,
↪  604328");
INSERT INTO Faculty VALUES("GECT","8795","Aswathy","MTECH",20,"CE","DFGJHH,
↪  634308");
INSERT INTO Faculty VALUES("GECT","8766","Arun","MTECH",16,"ECE","DFGJWU,
↪  634358");

/*Query (a)*/
SELECT
↪  F.facultyname,F.facultycode,F.qualification,F.collegecode,F.department
FROM Faculty AS F,College AS C
WHERE F.collegecode = "KTE" AND F.experience >= 10 AND F.collegecode =
↪  C.collegecode;

/*Query (b)*/
SELECT
↪  F.facultyname,F.facultycode,F.qualification,F.collegecode,F.department
FROM Faculty AS F, College AS C
WHERE F.collegecode = "KTE" AND F.experience >= 10 AND F.qualification !=
↪  "MTECH" AND F.collegecode = C.collegecode;

/*Query (c)*/
SELECT
↪  F.facultyname,F.facultycode,F.qualification,F.collegecode,F.department
FROM Faculty AS F,College AS C
WHERE F.collegecode = "KTE" AND F.collegecode = C.collegecode
ORDER BY F.department,F.experience ASC;

/*Query (d)*/
SELECT COUNT(*) , C.collegename
FROM Faculty AS F, College AS C
WHERE F.collegecode = C.collegecode
GROUP BY F.collegecode
HAVING (COUNT(*)>3);

/*Query (e)*/
SELECT collegename,MAX(empCount) AS  'empCount'
FROM (
    SELECT C.collegename,F.collegecode,COUNT(*) AS 'empCount'
    FROM Faculty as F,College as C
    WHERE F.collegecode = C.collegecode
    GROUP BY F.collegecode
    ORDER BY empCount DESC
```

```
    ) AS max
UNION
SELECT collegename,MIN(empCount) AS  'empCount'
FROM (
    SELECT C.collegename,F.collegecode,COUNT(*) AS 'empCount'
    FROM Faculty as F,College as C
    WHERE F.collegecode = C.collegecode
    GROUP BY F.collegecode
    ORDER BY empCount ASC
    )AS min;

DROP TABLE Faculty;
DROP TABLE College;
DROP DATABASE campus;
```

Output

```
+-------------+------------+--------------+-------------+------------+
| facultyname | facultycode | qualification | collegecode | department |
+-------------+------------+--------------+-------------+------------+
| Ajitha      | 8924       | MTECH        | KTE         | EEE        |
| Swathi      | 9004       | BTECH        | KTE         | EEE        |
| Priya       | 9087       | MTECH        | KTE         | CSE        |
+-------------+------------+--------------+-------------+------------+
3 rows in set (0.00 sec)

+-------------+------------+--------------+-------------+------------+
| facultyname | facultycode | qualification | collegecode | department |
+-------------+------------+--------------+-------------+------------+
| Swathi      | 9004       | BTECH        | KTE         | EEE        |
+-------------+------------+--------------+-------------+------------+
1 row in set (0.00 sec)

+-------------+------------+--------------+-------------+------------+
| facultyname | facultycode | qualification | collegecode | department |
+-------------+------------+--------------+-------------+------------+
| Ajith       | 9987       | BTECH        | KTE         | CSE        |
| Priya       | 9087       | MTECH        | KTE         | CSE        |
| Swathi      | 9004       | BTECH        | KTE         | EEE        |
| Ajitha      | 8924       | MTECH        | KTE         | EEE        |
+-------------+------------+--------------+-------------+------------+
4 rows in set (0.00 sec)

+----------+-------------+
| COUNT(*) | collegename  |
+----------+-------------+
|        4 | RIT,Kottayam |
+----------+-------------+
1 row in set (0.00 sec)

+-------------+----------+
| collegename  | empCount |
+-------------+----------+
| RIT,Kottayam |        4 |
| GEC,Wayanad  |        1 |
+-------------+----------+
2 rows in set (0.00 sec)
```

Figure 1: Output of Experiment 1

Experiment 2

Library Database
Date:-14/10/2022

Question:- Create the following tables for a Library Management System.
Book: consisting of fields accession-no, title, publisher, author, date-of-purchase, date-of-publishing, status
Status can be "issued", "present in the Library", "reference", "cannot be issued". Write SQL Queries for the following.
(a) List out the total number of copies of each book in the library.
(b) List out the total number of reference copies for each book in the Library.
(c) For each book in the Library obtain a count of the total number of issued copies, number copies existing at present in the library and the number of reference copies.
(d) List out the details of various books of each publisher with status of the books set to "cannot be issued".
(e) List out the details of the books which are new arrivals. The books which are purchased during the last 6 months are categorized as new arrivals.
(f) List out the details of each famous book . Each Famous book should be purchased within 1 year of its date-of –publishing and the number of total copies is more than 10.

Answer:-
The SQL script for solving the given question is the following.

```
CREATE DATABASE Library;
USE Library;

CREATE TABLE BOOK(
        accession_no int,
        title varchar(20),
        publisher varchar(30),
        author varchar(25),
        date_of_purchase DATE,
        date_of_publishing DATE,
        status varchar(20),
        PRIMARY KEY(accession_no)
);

INSERT INTO
 ↪  BOOK(accession_no,title,publisher,author,date_of_purchase,date_of_publishing,statu
VALUES
(001,'Computer Networks','Thakur Publications','Saurabh
 ↪  Singhal','2015-06-10','2015-05-19','issued'),
(002,'Computer Networks','Sun India Publications','Subhash Kumar
 ↪  Verma','2015-05-10','2009-10-06','Cannot be issued'),
(003,'Data
 ↪  Structures','McGraw','Lipschute','2017-07-02','2007-10-04','present in
 ↪  library'),
(004,'Turbo C++','Galgotia','Robort
 ↪  Lafore','2018-08-27','2003-09-17','issued'),
```

```
(005,'Basic for
↪  Beginners','BPB','Norton','2018-08-27','2008-07-12','present in
↪  library'),
(006,'Computer
↪  Studies','Galgotia','French','2018-02-04','2010-06-16','reference'),
(007,'Mastering
↪  Windows','BPB','Cowart','2005-07-22','2005-04-15','reference'),
(008,'Guide Network','Zpress','Freed','2017-07-02','2009-03-19','issued'),
(009,'Advanced Pascal','McGraw','Schildt','2018-08-27','2010-06-04','Cannot
↪  be issued'),
(010,'Dbase Dummies','PustakM','Palmer','2022-08-04','2008-07-18','present
↪  in library'),
(011,'Data
↪  Structures','McGraw','Lipschute','2008-07-02','2007-10-04','issued'),
(012,'Data
↪  Structures','McGraw','Lipschute','2017-07-02','2007-10-04','reference'),
(013,'Mastering Windows','BPB','Cowart','2005-07-22','2005-04-15','present
↪  in library'),
(014,'Computer
↪  Studies','Galgotia','French','2018-02-04','2010-06-16','reference'),
(015,'Computer
↪  Studies','Galgotia','French','2018-02-04','2010-06-16','reference'),
(016,'Mastering
↪  Windows','BPB','Cowart','2005-07-22','2005-04-15','reference'),
(017,'Mastering
↪  Windows','BPB','Cowart','2005-07-22','2005-04-15','issued'),
(018,'Mastering Windows','BPB','Cowart','2005-07-22','2005-04-15','present
↪  in library'),
(019,'Mastering Windows','BPB','Cowart','2005-07-22','2005-04-15','present
↪  in library'),
(020,'Advanced Pascal','McGraw','Schildt','2018-08-27','2010-06-04','Cannot
↪  be issued'),
(021,'Mastering
↪  Windows','BPB','Cowart','2005-07-22','2005-04-15','reference'),
(022,'Mastering Windows','BPB','Cowart','2005-07-22','2005-04-15','Cannot
↪  be issued'),
(023,'Mastering
↪  Windows','BPB','Cowart','2005-07-22','2005-04-15','reference'),
(024,'Mastering
↪  Windows','BPB','Cowart','2005-07-22','2005-04-15','issued'),
(025,'Mastering
↪  Windows','BPB','Cowart','2005-07-22','2005-04-15','reference');

/*Query (a)*/
SELECT title,author,publisher,count(accession_no)
FROM BOOK
GROUP BY title,author,publisher;

/*Query (b)*/
```

```
SELECT title,author,publisher,count(accession_no)
FROM BOOK
WHERE status ='reference'
GROUP BY title,author,publisher;

/*Query (c)*/
SELECT title,author,publisher,status,count(accession_no)
FROM BOOK
WHERE status ='reference' OR status ='issued' OR status ='present in
 ↪  library'
GROUP BY title,author,publisher,status;

/*Query (d)*/
SELECT *
FROM BOOK
WHERE status ='cannot be issued';

/*Query (e)*/
SELECT
 ↪  title,author,publisher,date_of_purchase,date_of_publishing,DATEDIFF(CURDATE(),date
FROM BOOK
GROUP BY title,author,publisher
HAVING DATEDIFF(CURDATE(),date_of_purchase) < 180;

/*Query (f)*/
SELECT title,author,publisher,date_of_purchase,date_of_publishing
FROM BOOK
GROUP BY title,author,publisher
HAVING DATEDIFF(date_of_purchase,date_of_publishing) < 365 and
 ↪  count(accession_no) > 10;

DROP TABLE BOOK;
DROP DATABASE Library;
```

Output

```
+--------------------+--------------------+-----------------------+--------------------+
| title              | author             | publisher             | count(accession_no) |
+--------------------+--------------------+-----------------------+--------------------+
| Computer Networks  | Saurabh Singhal    | Thakur Publications   |                  1 |
| Computer Networks  | Subhash Kumar Verma| Sun India Publications|                  1 |
| Data Structures    | Lipschute          | McGraw                |                  3 |
| Turbo C++          | Robort Lafore      | Galgotia              |                  1 |
| Basic for Beginners| Norton             | BPB                   |                  1 |
| Computer Studies   | French             | Galgotia              |                  3 |
| Mastering Windows  | Cowart             | BPB                   |                 11 |
| Guide Network      | Freed              | Zpress                |                  1 |
| Advanced Pascal    | Schildt            | McGraw                |                  2 |
| Dbase Dummies      | Palmer             | PustakM               |                  1 |
+--------------------+--------------------+-----------------------+--------------------+
10 rows in set (0.01 sec)
```

```
+-------------------+-----------+-----------+--------------------+
| title             | author    | publisher | count(accession_no) |
+-------------------+-----------+-----------+--------------------+
| Computer Studies  | French    | Galgotia  |                  3 |
| Mastering Windows | Cowart    | BPB       |                  5 |
| Data Structures   | Lipschute | McGraw    |                  1 |
+-------------------+-----------+-----------+--------------------+
3 rows in set (0.00 sec)

+--------------------+----------------+--------------------+-------------------+--------------------+
| title              | author         | publisher          | status            | count(accession_no) |
+--------------------+----------------+--------------------+-------------------+--------------------+
| Computer Networks  | Saurabh Singhal | Thakur Publications | issued            |                  1 |
| Data Structures    | Lipschute      | McGraw             | present in library |                  1 |
| Turbo C++          | Robort Lafore  | Galgotia           | issued            |                  1 |
| Basic for Beginners | Norton         | BPB                | present in library |                  1 |
| Computer Studies   | French         | Galgotia           | reference         |                  3 |
| Mastering Windows  | Cowart         | BPB                | reference         |                  5 |
| Guide Network      | Freed          | Zpress             | issued            |                  1 |
| Dbase Dummies      | Palmer         | PustakM            | present in library |                  1 |
| Data Structures    | Lipschute      | McGraw             | issued            |                  1 |
| Data Structures    | Lipschute      | McGraw             | reference         |                  1 |
| Mastering Windows  | Cowart         | BPB                | present in library |                  3 |
| Mastering Windows  | Cowart         | BPB                | issued            |                  2 |
+--------------------+----------------+--------------------+-------------------+--------------------+
12 rows in set (0.00 sec)

+-------------+------------------+----------------------+--------------------+-----------------+-------------------+-----------------+
| accession_no | title            | publisher            | author             | date_of_purchase | date_of_publishing | status          |
+-------------+------------------+----------------------+--------------------+-----------------+-------------------+-----------------+
|           2 | Computer Networks | Sun India Publications | Subhash Kumar Verma | 2015-05-10      | 2009-10-06        | Cannot be issued |
|           9 | Advanced Pascal   | McGraw               | Schildt            | 2018-08-27      | 2010-06-04        | Cannot be issued |
|          20 | Advanced Pascal   | McGraw               | Schildt            | 2018-08-27      | 2010-06-04        | Cannot be issued |
|          22 | Mastering Windows | BPB                  | Cowart             | 2005-07-22      | 2005-04-15        | Cannot be issued |
+-------------+------------------+----------------------+--------------------+-----------------+-------------------+-----------------+
4 rows in set (0.00 sec)

+--------------+--------+----------+-----------------+-------------------+-----------------------------------+
| title        | author | publisher | date_of_purchase | date_of_publishing | DATEDIFF(CURDATE(),date_of_purchase) |
+--------------+--------+----------+-----------------+-------------------+-----------------------------------+
| Dbase Dummies | Palmer | PustakM  | 2022-08-04      | 2008-07-18        |                                94 |
+--------------+--------+----------+-----------------+-------------------+-----------------------------------+
1 row in set (0.01 sec)

+-------------------+--------+----------+-----------------+-------------------+
| title             | author | publisher | date_of_purchase | date_of_publishing |
+-------------------+--------+----------+-----------------+-------------------+
| Mastering Windows | Cowart | BPB      | 2005-07-22      | 2005-04-15        |
+-------------------+--------+----------+-----------------+-------------------+
```

Figure 2: Output of Experiment 2

Experiment 3

Student Database
Date:-21/10/2022

Question:- Create the following tables.

Student (roll-on, name, date-of-birth)

Course (course-id, name, fee, duration)

Generate the Queries for the following.

(a) List the names of all students who are greater than 18 years of age and have opted B.Tech Course.

(b) List the details of those courses whose fee is greater than that of B.Tech Course.

(c) List the details of the students who have opted more than 2 courses.

(d) List the details (name, fee and duration) of the course which have been opted by maximum number of students and those of the course which is opted by the least number of students.

(e) List the details of the student(s) who have opted every course

Answer:-

The SQL script for solving the given question is the following.

```
create database college;
use college;


create table course(
        course_id varchar(15) not null primary key,
        name varchar(15),
        fee int,
        duration int
);

create table student(
        roll_no int not null,
        name varchar(20),
        date_of_birth date,
        course_id varchar(15) not null,
        foreign key (course_id) references course(course_id)
);


insert into course values("1","B.Tech",50000,4);
insert into course values("2","M.Tech",80000,2);
insert into course values("3","BCA",60000,3);
insert into course values("4","MCA",45000,2);

insert into student values(22,"Aswathy","2002-10-21","1");
insert into student values(22,"Aswathy","2002-10-21","2");
insert into student values(40,"Josmy","2002-08-30","1");
insert into student values(40,"Josmy","2002-08-30","2");
```

```
insert into student values(40,"Josmy","2002-08-30","4");
insert into student values(50,"Naveena","2002-05-26","2");
insert into student values(50,"Naveena","2002-05-26","3");
insert into student values(50,"Naveena","2002-05-26","4");
insert into student values(50,"Naveena","2002-05-26","1");
insert into student values(21,"Aswathi","2001-05-23","4");
insert into student values(35,"Minu","2006-08-24","1");

/*Query (a)*/
select S.name
from student as S, course as C
where S.course_id = C.course_id and S.course_id = "1" and
↪  year(curdate())-year(S.date_of_birth)<18;

/*Query (b)*/
select d.course_id,d.name,d.fee,d.duration
from course as c,course as d
where d.fee > c.fee and c.course_id = "1";

/*Query (c)*/
select roll_no,name,date_of_birth
from student
group by roll_no,name,date_of_birth
having count(course_id)>2;

/*Query (d)*/
SELECT name,fee,duration,MAX( studentCount ) AS  'studentCount'
FROM (
    SELECT C.name,C.fee,C.duration,COUNT(S.course_id) AS  'studentCount'
    FROM student as S, course as C
    where S.course_id = C.course_id
    GROUP BY S.course_id
    ORDER BY studentCount DESC
    ) AS max
UNION
SELECT name,fee,duration,MIN( studentCount ) AS  'studentCount'
FROM (
    SELECT C.name,C.fee,C.duration,COUNT(S.course_id) AS  'studentCount'
    FROM student as S, course as C
    where S.course_id = C.course_id
    GROUP BY S.course_id
    ORDER BY studentCount ASC
    )AS min;

/*Query (e)*/
select roll_no,name,date_of_birth
from student
group by roll_no,name,date_of_birth
having count(course_id)=4;
```

```
drop table student;
drop table course;
drop database college;
```

Output

```
+------+
| name |
+------+
| Minu |
+------+
1 row in set (0.01 sec)

+-----------+--------+-------+----------+
| course_id | name   | fee   | duration |
+-----------+--------+-------+----------+
| 2         | M.Tech | 80000 |        2 |
| 3         | BCA    | 60000 |        3 |
+-----------+--------+-------+----------+
2 rows in set (0.00 sec)

+---------+---------+---------------+
| roll_no | name    | date_of_birth |
+---------+---------+---------------+
|      40 | Josmy   | 2002-08-30    |
|      50 | Naveena | 2002-05-26    |
+---------+---------+---------------+
2 rows in set (0.00 sec)

+--------+-------+----------+--------------+
| name   | fee   | duration | studentCount |
+--------+-------+----------+--------------+
| B.Tech | 50000 |        4 |            4 |
| BCA    | 60000 |        3 |            1 |
+--------+-------+----------+--------------+
2 rows in set (0.00 sec)

+---------+---------+---------------+
| roll_no | name    | date_of_birth |
+---------+---------+---------------+
|      50 | Naveena | 2002-05-26    |
+---------+---------+---------------+
```

Figure 3: Output of Experiment 3

Experiment 4

Result Database
Date:-28/10/2022

Question:- Create the following tables.

Student(rollno, name, category, district, state)

Student-rank (rollno, mark, rank)

Generate Queries for the following.

(a) List the details of the students with the same category and same rank.

(b) List out the details of the students (rollno, name, category, district, rank) who secured the highest rank for each category in each state.

(c) List the names of the students(roll no, name, category, district, mark, rank) having either marks same but ranks different or marks different but ranks same together with the status (whether they belong to the first category or second category)

(d) Find the category with the highest academic performance and the one with the least academic performance.

(e) Find the category whose academic performance is below the average academic performance.

Answer:-

The SQL script for solving the given question is the following.

```
create database Student_details;
use Student_details;

create table student(
rollno int not null primary key,
name varchar(30),
category varchar(10),
district varchar(20),
state varchar(20)
);

create table student_rank(
rollno int not null,
mark int,
erank int,
foreign key(rollno) references student(rollno)
);

INSERT INTO student(rollno, name, category, district, state)
VALUES
(001,'Hrithik','OBC','Ballari','Karnataka'),
(002,'Himanshi','OBC','Chitradurga','Karnataka'),
(003,'Angel','SC','Bangalore','Karnataka'),
(004,'Harris','GENERAL','Tumkur','Karnataka'),
(005,'Fathima','GENERAL','Raichur','Karnataka'),
(006,'Meena','OBC','Bagalkot','Karnataka'),
(007,'Glory','SC','Ballari','Karnataka'),
```

```
(008,'Ann Mary','SC','Ernakulam','Kerala'),
(009,'Ancilla','GENERAL','Kottayam','Kerala'),
(010,'Alex','SC','Ernakulam','Kerala'),
(011,'Aadhil','GENERAL','Malappuram','Kerala'),
(012,'Sana','GENERAL','Kozhikode','Kerala'),
(013,'Aravind','OBC','Thrissur','Kerala'),
(014,'Gayathri','OBC','Kollam','Kerala'),
(015,'Aleena','GENERAL','Ernakulam','Kerala');

INSERT INTO student_rank(rollno,mark,erank)
VALUES
(001,256,15),
(002,392,54),
(003,473,3),
(004,402,7),
(005,305,13),
(006,406,54),
(007,419,6),
(008,389,46),
(009,498,1),
(010,473,2),
(011,387,10),
(012,434,4),
(013,285,14),
(014,344,12),
(015,389,46);

/*Query(a)*/
create view Temp1
as
select *
from student
natural join student_rank;
select *
from Temp1;
select S1.*
from Temp1 as S1,Temp1 as S2
where S1.category=S2.category and S1.erank =S2.erank and S1.rollno!=
↪  S2.rollno
ORDER BY S1.category,S1.erank;

/*Query(b)*/
select s.rollno,s.name,s.category,s.district,s.state,min(t.erank) as
↪  'minrank'
from student as s, student_rank as t
where s.rollno = t.rollno
group by s.state,s.category
order by minrank ASC;
```

```
/*Query(c)*/
create table T_Status(
    Status varchar(30)
);
insert into T_Status values
("same mark different rank"),
("same rank different mark");
create view same_mark as
select distinct s.name as Student_1,s.erank as Rank_1,s.mark as
 ↪  Mark_1,Status
from Temp1 as s,Temp1 as t,T_Status
where s.mark=t.mark and s.erank!=t.erank and Status="same mark different
 ↪  rank"
order by s.mark;
create view same_rank as
select distinct s.name as Student_1,s.erank as Rank_1,s.mark as
 ↪  Mark_1,Status
from Temp1 as s,Temp1 as t,T_Status
where s.erank=t.erank and s.mark!=t.mark and Status="same rank different
 ↪  mark"
order by s.erank;
select * from same_mark
union
select * from same_rank;
drop view same_mark;
drop view same_rank;

/*Query(d)*/
select category,max(avgac) as 'avgac'
from(select s.category,avg(t.mark) as 'avgac'
     from student as s, student_rank as t
     where s.rollno = t.rollno
     group by s.category
     order by avgac DESC)as maxm
UNION
select category,min(avgac) as 'avgac'
from(select s.category,avg(t.mark) as 'avgac'
     from student as s, student_rank as t
     where s.rollno = t.rollno
     group by s.category
     order by avgac ASC)as minm;

/*Query(e)*/
select category,min(avgac) as 'avgac'
from (select s.category,avg(t.mark) as 'avgac'
      from student as s, student_rank as t
      where s.rollno = t.rollno
      group by s.category
      order by avgac ASC)AS minm;
```

```
drop view Temp1;
drop table student_rank;
drop table student;
drop database Student_details;
```

Output

```
+--------+----------+----------+------------+-----------+------+-------+
| rollno | name     | category | district   | state     | mark | erank |
+--------+----------+----------+------------+-----------+------+-------+
|      2 | Himanshi | OBC      | Chitradurga | Karnataka |  392 |    54 |
|      6 | Meena    | OBC      | Bagalkot   | Karnataka |  406 |    54 |
+--------+----------+----------+------------+-----------+------+-------+
2 rows in set (0.00 sec)

+--------+----------+----------+-----------+-----------+---------+
| rollno | name     | category | district  | state     | minrank |
+--------+----------+----------+-----------+-----------+---------+
|      9 | Ancilla  | GENERAL  | Kottayam  | Kerala    |       1 |
|      8 | Ann Mary | SC       | Ernakulam | Kerala    |       2 |
|      3 | Angel    | SC       | Bangalore | Karnataka |       3 |
|      4 | Harris   | GENERAL  | Tumkur    | Karnataka |       7 |
|     13 | Aravind  | OBC      | Thrissur  | Kerala    |      12 |
|      1 | Hrithik  | OBC      | Ballari   | Karnataka |      15 |
+--------+----------+----------+-----------+-----------+---------+
6 rows in set (0.00 sec)
```

```
+-----------+--------+--------+--------------------------+
| Student_1 | Rank_1 | Mark_1 | Status                   |
+-----------+--------+--------+--------------------------+
| Angel     |      3 |    473 | same mark different rank |
| Alex      |      2 |    473 | same mark different rank |
| Himanshi  |     54 |    392 | same rank different mark |
| Meena     |     54 |    406 | same rank different mark |
+-----------+--------+--------+--------------------------+
4 rows in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

+----------+----------+
| category | avgac    |
+----------+----------+
| SC       | 438.5000 |
| OBC      | 336.6000 |
+----------+----------+
2 rows in set (0.00 sec)

+----------+----------+
| category | avgac    |
+----------+----------+
| OBC      | 336.6000 |
+----------+----------+
1 row in set (0.00 sec)
```

Figure 4: Output of Experiment 4

Experiment 5

Book Issue Database
Date:-09/11/2022

Question:- Create the following tables.

Book(accession-no, title, publisher, year, date-of-purchase, status)

Member(member-id, name, number-of-books-issued, max-limit)

Books- issue(accession-no, member-id,date-of-issue)

Generate SQL Queries for the following.

(a) List all those books which are due from the students. A Book is considered as Due if it has been issued 15 days back and not yet returned.

(b) List all members who cannot be issued any more books.

(c) List the details of the book which is taken by the maximum number of members and the book which is taken by the least number of members.

(d) List the details of the book which is taken by every member and the one that is not yet issued.

Answer:-

The SQL script for solving the given question is the following.

```
create database library;
use library;

create table book(
accession_no int primary key,
title varchar(40),
publisher varchar(40),
year_of_publishing YEAR,
date_of_purchase date,
status varchar(20)
);
create table member(
member_id int primary key,
name varchar(30),
number_of_books_issued int,
max_limit int
);
create table books_issue(
accession_no int,
member_id int,
date_of_issue date,
foreign key (accession_no) references book(accession_no),
foreign key (member_id) references member(member_id)
);

INSERT INTO book VALUES
("122","MIDNIGHT CHILDREN","SALMAN RUSHDIE","1947","2002-05-18","ISSUED"),
("123","THE MAGIC MOUNTAIN","THOMAS MANN","1998","2003-05-28","ISSUED"),
```

```
("124","GREAT EXPECTATIONS","CHARLES
↪  DICKENS","2005","2008-01-28","ISSUED"),
("125","LEAVES OF GRASS","WALT WHITMAN","2008","2011-11-26","ISSUED"),
("126","TRISTRAM SHANDY","LAURENCE STRENE","1996","2002-01-08","ISSUED"),
("127","DAVID COPPERFIELD","CHARLES DICKENS","1999","2003-02-07","ISSUED"),
("128","THE AENEID","VIRGIL","2000","2001-12-10","ISSUED"),
("129","JANE EYRE","CHARLOTTE BRONTE","2010","2011-04-05","ISSUED"),
("130","THE STRANGER","ALBERT CAMUS","2021","2022-04-12","ISSUED"),
("131","BELOVED","TONI MORRISON","2004","2021-07-22","ISSUED"),
("132","MIDDLEMARCH","GEORGE ELIOT","2002","2004-10-21","ISSUED"),
("133","INVISIBLE MAN","RALPH ELLISON","2006","2007-10-10","ISSUED"),
("134","MIDNIGHT CHILDREN","SALMAN RUSHDIE","1947","2002-05-18","NOT
↪  ISSUED"),
("135","THE MAGIC MOUNTAIN","THOMAS MANN","1998","2003-05-28","NOT
↪  ISSUED"),
("136","GREAT EXPECTATIONS","CHARLES DICKENS","2005","2008-01-28","NOT
↪  ISSUED"),
("137","LEAVES OF GRASS","WALT WHITMAN","2008","2011-11-26","NOT ISSUED"),
("138","TRISTRAM SHANDY","LAURENCE STRENE","1996","2002-01-08","NOT
↪  ISSUED"),
("139","DAVID COPPERFIELD","CHARLES DICKENS","1999","2003-02-07","NOT
↪  ISSUED"),
("140","THE AENEID","VIRGIL","2000","2001-12-10","NOT ISSUED"),
("141","JANE EYRE","CHARLOTTE BRONTE","2010","2011-04-05","NOT ISSUED"),
("142","THE STRANGER","ALBERT CAMUS","2021","2022-04-12","NOT ISSUED"),
("143","BELOVED","TONI MORRISON","2004","2021-07-22","NOT ISSUED"),
("144","MIDDLEMARCH","GEORGE ELIOT","2002","2004-10-21","NOT ISSUED"),
("145","INVISIBLE MAN","RALPH ELLISON","2006","2007-10-10","NOT ISSUED"),
("146","MIDNIGHT CHILDREN","SALMAN
↪  RUSHDIE","1947","2002-05-18","REFERENCE"),
("147","THE MAGIC MOUNTAIN","THOMAS MANN","1998","2003-05-28","REFERENCE"),
("148","GREAT EXPECTATIONS","CHARLES
↪  DICKENS","2005","2008-01-28","REFERENCE"),
("149","LEAVES OF GRASS","WALT WHITMAN","2008","2011-11-26","REFERENCE"),
("150","TRISTRAM SHANDY","LAURENCE
↪  STRENE","1996","2002-01-08","REFERENCE"),
("151","DAVID COPPERFIELD","CHARLES
↪  DICKENS","1999","2003-02-07","REFERENCE"),
("152","THE AENEID","VIRGIL","2000","2001-12-10","REFERENCE"),
("153","JANE EYRE","CHARLOTTE BRONTE","2010","2011-04-05","REFERENCE"),
("154","THE STRANGER","ALBERT CAMUS","2021","2022-04-12","REFERENCE"),
("155","BELOVED","TONI MORRISON","2004","2021-07-22","REFERENCE"),
("156","MIDDLEMARCH","GEORGE ELIOT","2002","2004-10-21","REFERENCE"),
("157","INVISIBLE MAN","RALPH ELLISON","2006","2007-10-10","REFERENCE");

INSERT INTO member VALUES
("2010","JOHN WICK","3","5"),
("2011","MONKEY D LUFFY","2","6"),
("2012","TONY STARK","8","8"),
```

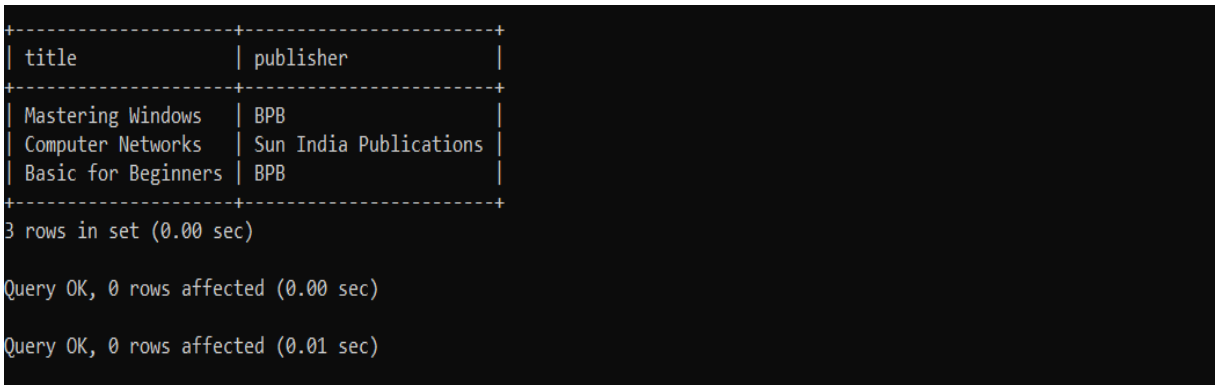
```
("2013","STEVEN GRANT ROGERS","1","2"),
("2014","CLINTON FRANCIS BARTON","2","7"),
("2015","WANDA MAXIMOFF","4","5"),
("2016","T CHALLA","1","5"),
("2017","VICTOR SHADE","4","4"),
("2018","THOR ODINSON","0","1"),
("2019","NATASHA ROMANOFF","4","5"),
("2020","JENNIFER SUSAN","4","5");

INSERT INTO books_issue VALUES
("122","2010","2005-11-18"),
("123","2010","2022-11-18"),
("124","2010","2022-11-18"),
("125","2011","2022-11-18"),
("126","2011","2022-11-18"),
("127","2012","2022-11-18"),
("128","2012","2011-11-18"),
("129","2012","2022-11-18"),
("130","2012","2022-11-18"),
("131","2012","2021-11-18"),
("132","2012","2022-11-18"),
("133","2012","2022-11-18"),
("122","2012","2022-11-18"),
("123","2013","2022-11-18"),
("124","2014","2022-11-18"),
("125","2014","2022-11-18"),
("126","2015","2022-11-18"),
("127","2015","2021-11-18"),
("128","2015","2022-11-18"),
("129","2015","2022-11-18"),
("130","2016","2022-11-18"),
("131","2017","2022-11-18"),
("132","2017","2022-11-18"),
("133","2017","2022-11-18"),
("122","2017","2022-11-18"),
("123","2017","2022-11-18"),
("124","2019","2022-11-18"),
("125","2019","2020-11-18"),
("126","2019","2022-11-18"),
("127","2019","2022-11-18"),
("128","2020","2022-11-18"),
("129","2020","2022-11-18"),
("130","2020","2022-11-18"),
("132","2020","2022-11-18");

select b.accession_no,b.title
from book as b, books_issue as c
where b.accession_no = c.accession_no and
 ↪  DATEDIFF(CURDATE(),c.date_of_issue)>15;
```

```
select m.member_id,m.name,m.number_of_books_issued
from member as m
where m.number_of_books_issued = m.max_limit;

create view temp4
as
select *
from book
natural join books_issue;

create view tempo
as
SELECT S.title,S.publisher,COUNT(*) AS  'bookCount'
FROM book as S
where S.status = "ISSUED"
GROUP BY S.title,S.publisher;
create view tempu
as
SELECT S.title,S.publisher,COUNT(*) AS  'bookCount'
FROM book as S
GROUP BY S.title,S.publisher;

create view temp1
as
select max(bookCount) as 'maxm'
from tempo;
create view temp2
as
select min(bookCount) as 'minm'
from tempo;

select b.title,b.publisher,count(i.member_id)
from book as b,books_issue as i
where b.accession_no = i.accession_no
group by b.title,b.publisher
having count(i.member_id) =  (select maxm
             from temp1)
union
select b.title,b.publisher,count(i.member_id)
from book as b,books_issue as i
where b.accession_no = i.accession_no
group by b.title,b.publisher
having count(i.member_id) =  (select minm
             from temp2);

SELECT b.title,b.publisher
FROM book as b,books_issue as t
where b.accession_no = t.accession_no
```

```
GROUP BY b.title,b.publisher
having count(*)=11
union
select title,publisher
from book
where status = "NOT ISSUED" or status = "REFERENCE"
group by title,publisher
having count(*) = (select bookCount
  from tempu
  where book.title = title and book.publisher = publisher);

drop view tempu;
drop view tempo;
drop view temp4;
drop view temp1;
drop view temp2;
drop table books_issue;
drop table member;
drop table book;
drop database library;
```

Output

```
+--------------+-------------------+
| accession_no | title             |
+--------------+-------------------+
|           17 | Mastering Windows |
|           24 | Mastering Windows |
|           15 | Computer Studies  |
|           11 | Data Structures   |
|            1 | Computer Networks |
|            4 | Turbo C++         |
|            8 | Guide Network     |
|           13 | Mastering Windows |
|           20 | Advanced Pascal   |
|            9 | Advanced Pascal   |
|           25 | Mastering Windows |
|           21 | Mastering Windows |
|           10 | Dbase Dummies     |
|           16 | Mastering Windows |
+--------------+-------------------+
14 rows in set (0.00 sec)

+-----------+---------+------------------------+
| member_id | name    | number_of_books_issued |
+-----------+---------+------------------------+
|       202 | Naveena |                      4 |
+-----------+---------+------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

+-------------------+--------------------+--------------------+
| title             | publisher          | count(i.member_id) |
+-------------------+--------------------+--------------------+
| Mastering Windows | BPB                |                  6 |
| Computer Studies  | Galgotia           |                  1 |
| Data Structures   | McGraw             |                  1 |
| Computer Networks | Thakur Publications |                  1 |
| Turbo C++         | Galgotia           |                  1 |
| Guide Network     | Zpress             |                  1 |
| Dbase Dummies     | PustakM            |                  1 |
+-------------------+--------------------+--------------------+
7 rows in set (0.01 sec)
```



```
+---------------------+-----------------------+
| title               | publisher             |
+---------------------+-----------------------+
| Mastering Windows   | BPB                   |
| Computer Networks   | Sun India Publications |
| Basic for Beginners | BPB                   |
+---------------------+-----------------------+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

Figure 5: Output of Experiment 5

Experiment 6

Bank Database
Date:-18/11/2022

Question:- Create the following tables.

Branch(branch-id, branch-name, branch-city)

Customer(customer-id, customer-name, customer-city)

Savings(customer-id, branch-id, Saving-accno, balance)

Loan(customer-id, branch-id, loan-accno, balance)

Generate the Queries for the following.

(a) List out the details of the customers who live in the same city as they have savings or loan account.

(b) List out the customers who have an account in a given branch-city.

(c) List out the customers who have an account in more than one branch.

(d) List out details of the customer who have

i. neither a saving account but a loan

ii. neither a loan but has a saving account.

iii. having both loan and saving.

(e) List the names of the customers who have no saving at all but having loan in more than two branches.

(f) For each branch produce a list of the total number of customers, total number of customers with loan only, total number of customers with saving only and the total number of customers with both loan and saving.

(g) Find the details of the branch which has issued max amount of loan.

(h) Find the details of the branch which has not yet issued any loan at all.

(i) For each customer produce a list consisting of the total saving balance, loan balance for those branches where he has either a loan or a saving account or both.

Answer:-

The SQL script for solving the given question is the following.

```
create database exp6;
use exp6;

create table Branch(
        branch_id int primary key,
        branch_name varchar(30),
        branch_city varchar(30)
);

create table Customer(
        customer_id int primary key,
        customer_name varchar(30),
        customer_city varchar(30)
);

create table Savings(
        customer_id int not NULL,
        branch_id int not NULL,
        saving_accno int,
```

```
        balance int,
        constraint fkc foreign key(customer_id) references
         ↪  Customer(customer_id),
        constraint fkb foreign key(branch_id) references Branch(branch_id)
);

create table Loan(
        customer_id int not NULL,
        branch_id int not NULL,
        loan_accno int,
        balance int,
        constraint fkc2 foreign key(customer_id) references
         ↪  Customer(customer_id),
        constraint fkb2 foreign key(branch_id) references Branch(branch_id)
);

insert into Branch values
(401,"Thiruvananthapuram","Thiruvananthapuram"),
(402,"Kottayam","Kottayam"),
(403,"Ernakulam","Kochi"),
(404,"Kozhikode","Kozhikode");

insert into Customer values
(6001,"Ananthakrishnan","Thiruvananthapuram"),
(6002,"Irfan","Thiruvananthapuram"),
(6003,"Suneeth","Thiruvananthapuram"),
(6004,"Sreejith","Kottayam"),
(6005,"Jafar","Kazhakoottam"),
(6006,"Radika","Pampady"),
(6007,"Jameela","Kanjikuzhi"),
(6008,"Bindu","Kottayam"),
(6009,"Purushothaman","Kollam"),
(6010,"Vincy","Kottayam"),
(6011,"Abdul Rahman","Thrissur"),
(6012,"Vishwanathan","Ernakulam"),
(6013,"Marykutty","Mattancheri"),
(6014,"Hajara","Ernakulam"),
(6015,"Revathy","Kozhikode"),
(6016,"Hameed","Perambra"),
(6017,"Suchithra","Kozhikode"),
(6018,"Saneesh","North Paravoor"),
(6019,"Gokul Das","Kozhikode"),
(6020,"Abraham","Kappad");

insert into Savings values
(6001,401,6400101,15000),
(6002,401,6400102,200000),
(6005,401,6400105,30000),
(6007,401,6400107,70000),
```

```
(6004,402,6400204,400000),
(6006,402,6400206,100000),
(6007,402,6400207,40000),
(6008,402,6400208,74000),
(6010,402,6400210,128507),
(6011,403,6400311,700000),
(6001,403,6400301,200000),
(6012,403,6400312,500000),
(6013,403,6400313,250000),
(6015,404,6400415,100000),
(6016,404,6400416,90756);

insert into Loan values
(6005,401,4600105,100000),
(6003,401,4600103,200000),
(6009,401,4600109,150000),
(6019,401,4600119,100000),
(6011,403,4600311,1000000),
(6013,403,4600313,500000),
(6014,403,4600314,300000),
(6018,403,4600318,300000),
(6006,403,4600306,100000),
(6009,403,4600309,200000),
(6014,404,4600414,100000),
(6016,404,4600416,150000),
(6017,404,4600417,200000),
(6019,404,4600419,300000),
(6020,404,4600419,400000),
(6008,404,4600408,100000);

create view AllSave as
select s.customer_id,s.branch_id,s.saving_accno as
↪  accno,s.balance,b.branch_name,b.branch_city,c.customer_name,c.customer_city
from Savings as s,Branch as b,Customer as c
where s.customer_id=c.customer_id and s.branch_id=b.branch_id;

create view AllLoan as
select s.customer_id,s.branch_id,s.loan_accno as
↪  accno,s.balance,b.branch_name,b.branch_city,c.customer_name,c.customer_city
from Loan as s,Branch as b,Customer as c
where s.customer_id=c.customer_id and s.branch_id=b.branch_id;

create view AllAcc as
(select * from AllSave
union
select * from AllLoan)
order by customer_id;

select customer_id,customer_name,customer_city,branch_city
```

```
from AllAcc
where customer_city=branch_city;

select distinct customer_id,customer_name,branch_city
from AllAcc
where branch_city="Kochi";

select distinct a1.customer_id,a1.customer_name
from AllAcc as a1,AllAcc as a2
where a1.customer_id=a2.customer_id and a1.branch_id!=a2.branch_id
order by a1.customer_id;


select distinct l.customer_id,l.customer_name as customer_loan_only
from AllLoan as l
where l.customer_id not in (select customer_id from AllSave)
order by l.customer_id;

select distinct l.customer_id,l.customer_name as customer_savings_only
from AllSave as l
where l.customer_id not in (select customer_id from AllLoan)
order by l.customer_id;

select distinct l.customer_id,l.customer_name as customer_loan_and_savings
from AllLoan as l,AllSave as s
where l.customer_id = s.customer_id
order by l.customer_id;

select l.customer_id,l.customer_name,count(*) as no_of_loans
from AllLoan as l
where l.customer_id not in (select customer_id from AllSave)
group by l.customer_id
having no_of_loans>1
order by l.customer_id;

create view c_loan as
select distinct l.branch_id, count(*) as no_l
from AllLoan as l
where l.customer_id not in (select s.customer_id from AllSave as s where
↪  l.branch_id=s.branch_id)
group by branch_id
order by l.branch_id;

create view c_save as
select distinct l.branch_id, count(*) as no_s
from AllSave as l
where l.customer_id not in (select s.customer_id from AllLoan as s where
↪  l.branch_id=s.branch_id)
group by branch_id
```

```
order by l.branch_id;

create view c_both as
select distinct l.branch_id, count(*) as no_bo
from AllLoan as l,AllSave as s
where l.customer_id = s.customer_id and l.branch_id=s.branch_id
group by branch_id
order by l.customer_id;

create view branch_details as
select b.branch_id,b.branch_name,
        ifnull(l.no_l,0) as no_of_customer_loan_only,
        ifnull(s.no_s,0) as no_of_customer_savings_only,
        ifnull(bo.no_bo,0) as no_of_customer_both,
        ifnull(l.no_l,0)+ifnull(s.no_s,0)+ifnull(bo.no_bo,0) as
        ↪  no_of_customers_total
from Branch as b
left join c_both as bo on b.branch_id=bo.branch_id
left join c_loan as l on b.branch_id=l.branch_id
left join c_save as s on b.branch_id=s.branch_id;

select * from branch_details;

create view t_loan as
select branch_id,branch_name,sum(balance) as total
from AllLoan group by branch_id;

select * from t_loan
where total in (select max(total) from t_loan);

select branch_id,branch_name,no_of_customer_loan_only
from branch_details
where no_of_customer_loan_only=0;

create view customer_details as
select c.customer_id,c.customer_name,ifnull(l.balance,0) as
↪  l_balance,ifnull(s.balance,0) as s_balance
from AllAcc as c
left join Loan as l on c.accno=l.loan_accno
left join Savings as s on c.accno=s.saving_accno;

select customer_id,customer_name,sum(l_balance) as
↪  loan_balance,sum(s_balance) as savings_balance
from customer_details
group by customer_id;

drop view AllAcc;
drop view AllLoan;
drop view AllSave;
```

```
drop table Loan;
drop table Savings;
drop table Customer;
drop table Branch;
drop database exp6;
```

Output

```
+-------------+----------------+-------------------+-------------------+
| customer_id | customer_name  | customer_city     | branch_city       |
+-------------+----------------+-------------------+-------------------+
|        6001 | Ananthakrishnan | Thiruvananthapuram | Thiruvananthapuram |
|        6002 | Irfan          | Thiruvananthapuram | Thiruvananthapuram |
|        6003 | Suneeth        | Thiruvananthapuram | Thiruvananthapuram |
|        6004 | Sreejith       | Kottayam          | Kottayam          |
|        6008 | Bindu          | Kottayam          | Kottayam          |
|        6010 | Vincy          | Kottayam          | Kottayam          |
|        6015 | Revathy        | Kozhikode         | Kozhikode         |
|        6017 | Suchithra      | Kozhikode         | Kozhikode         |
|        6019 | Gokul Das      | Kozhikode         | Kozhikode         |
+-------------+----------------+-------------------+-------------------+
9 rows in set (0.01 sec)

+-------------+----------------+-------------+
| customer_id | customer_name  | branch_city |
+-------------+----------------+-------------+
|        6001 | Ananthakrishnan | Kochi       |
|        6006 | Radika         | Kochi       |
|        6009 | Purushothaman  | Kochi       |
|        6011 | Abdul Rahman   | Kochi       |
|        6012 | Vishwanathan   | Kochi       |
|        6013 | Marykutty      | Kochi       |
|        6014 | Hajara         | Kochi       |
|        6018 | Saneesh        | Kochi       |
+-------------+----------------+-------------+
8 rows in set (0.00 sec)

+-------------+----------------+
| customer_id | customer_name  |
+-------------+----------------+
|        6001 | Ananthakrishnan |
|        6006 | Radika         |
|        6007 | Jameela        |
|        6008 | Bindu          |
|        6009 | Purushothaman  |
|        6014 | Hajara         |
|        6019 | Gokul Das      |
+-------------+----------------+
7 rows in set (0.01 sec)
```

```
+-------------+--------------------+
| customer_id | customer_loan_only |
+-------------+--------------------+
|        6003 | Suneeth            |
|        6009 | Purushothaman      |
|        6014 | Hajara             |
|        6017 | Suchithra          |
|        6018 | Saneesh            |
|        6019 | Gokul Das          |
|        6020 | Abraham            |
+-------------+--------------------+
7 rows in set (0.01 sec)

+-------------+-----------------------+
| customer_id | customer_savings_only |
+-------------+-----------------------+
|        6001 | Ananthakrishnan       |
|        6002 | Irfan                 |
|        6004 | Sreejith              |
|        6007 | Jameela               |
|        6010 | Vincy                 |
|        6012 | Vishwanathan          |
|        6015 | Revathy               |
+-------------+-----------------------+
7 rows in set (0.00 sec)

+-------------+--------------------------+
| customer_id | customer_loan_and_savings |
+-------------+--------------------------+
|        6005 | Jafar                    |
|        6006 | Radika                   |
|        6008 | Bindu                    |
|        6011 | Abdul Rahman             |
|        6013 | Marykutty                |
|        6016 | Hameed                   |
+-------------+--------------------------+
6 rows in set (0.00 sec)

+-------------+---------------+-------------+
| customer_id | customer_name | no_of_loans |
+-------------+---------------+-------------+
|        6009 | Purushothaman |           2 |
|        6014 | Hajara        |           2 |
|        6019 | Gokul Das     |           2 |
+-------------+---------------+-------------+
3 rows in set (0.00 sec)
```

```
+-----------+------------------+-----------------------+--------------------------+-------------------+----------------------+
| branch_id | branch_name      | no_of_customer_loan_only | no_of_customer_savings_only | no_of_customer_both | no_of_customers_total |
+-----------+------------------+-----------------------+--------------------------+-------------------+----------------------+
|       401 | Thiruvananthapuram |                     3 |                        3 |                 1 |                    7 |
|       402 | Kottayam         |                     0 |                        5 |                 0 |                    5 |
|       403 | Ernakulam        |                     4 |                        2 |                 2 |                    8 |
|       404 | Kozhikode        |                     5 |                        1 |                 1 |                    7 |
+-----------+------------------+-----------------------+--------------------------+-------------------+----------------------+
4 rows in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

+-----------+------------+---------+
| branch_id | branch_name | total   |
+-----------+------------+---------+
|       403 | Ernakulam  | 2400000 |
+-----------+------------+---------+
1 row in set (0.00 sec)

+-----------+------------+-----------------------+
| branch_id | branch_name | no_of_customer_loan_only |
+-----------+------------+-----------------------+
|       402 | Kottayam   |                     0 |
+-----------+------------+-----------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

```
+-------------+-----------------+--------------+-----------------+
| customer_id | customer_name   | loan_balance | savings_balance |
+-------------+-----------------+--------------+-----------------+
|        6001 | Ananthakrishnan |            0 |          215000 |
|        6002 | Irfan           |            0 |          200000 |
|        6003 | Suneeth         |       200000 |               0 |
|        6004 | Sreejith        |            0 |          400000 |
|        6005 | Jafar           |       100000 |           30000 |
|        6006 | Radika          |       100000 |          100000 |
|        6007 | Jameela         |            0 |          110000 |
|        6008 | Bindu           |       100000 |           74000 |
|        6009 | Purushothaman   |       350000 |               0 |
|        6010 | Vincy           |            0 |          128507 |
|        6011 | Abdul Rahman    |      1000000 |          700000 |
|        6012 | Vishwanathan    |            0 |          500000 |
|        6013 | Marykutty       |       500000 |          250000 |
|        6014 | Hajara          |       400000 |               0 |
|        6015 | Revathy         |            0 |          100000 |
|        6016 | Hameed          |       150000 |           90756 |
|        6017 | Suchithra       |       200000 |               0 |
|        6018 | Saneesh         |       300000 |               0 |
|        6019 | Gokul Das       |       800000 |               0 |
|        6020 | Abraham         |       700000 |               0 |
+-------------+-----------------+--------------+-----------------+
20 rows in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)
```

Figure 6: Output of Experiment 6

Experiment 7

College DB
Date:-09/12/2022

Question:- A student academic system to be maintained in an engineering college should have the following facilities.

(a) The system should keep

i. student details(student name, roll no, utyreg no, address, year of admin, year of pass out, branch of study, class teacher, emailid, phone no).

ii. the details of course qualification(total and the grade(first class, second class, distinction)).

iii. the details of project work done by each student(project title, project guide(a faculty from the college),period of implementation of the project, core area)

iv. the faculty details(faculty name, faculty id, emailid, designation, joining date, relieved date).

(b) The system must have the facility to give the answers to the following questions.

i. The names, roll no and address of the students who have completed the course under a given branch for each year.

ii. The names and roll nos of the students who completed the course for each year of pass out and for each branch.

iii. The details of projects under taken by the students of a particular branch.

iv. The name of the faculty who guided more than a specified no. of projects in each academic year.

v. The branch with highest academic performance, chosen for each academic year.

vi. The details of the students who secured the highest total for each branch and for each academic year.

vii. The list of students who secured a given grade, for a given academic year.

viii. The list of projects undertaken in each department together with the project guide name and emailid, for each academic year under a given core area.

ix. The number of total grades for each branch of study and for each year of admission.

x. The details of students in each branch admitted in a specified academic year.

xi. The details of the students, sorted on the basis of year of admission and branch of study.

xii. The best mark, worst mark and the avg mark for each branch for a given academic year.

Construct the ER diagram for this system. Maintain appropriate tables to keep the information specified in (a). Write the SQL queries for getting the results to the questions mentioned in (b).

Answer:-

The SQL script for solving the given question is the following.

```
CREATE DATABASE EXPT;
USE EXPT;

CREATE TABLE STU_DET(
stname varchar(50) ,
stroll int,
utyreg varchar(20) primary key,
addr varchar(50),
adm_year int,
pass_year int,
branch varchar(30),
cl_teach varchar(50),
e_mail varchar(50),
phno varchar(15)
```

```
);

CREATE TABLE QUALIF(
utyreg varchar(20),
total int,
grade varchar(15),
year varchar(5),
CONSTRAINT FOREIGN KEY f1(utyreg) REFERENCES STU_DET(utyreg) ON DELETE
 ↪  CASCADE ON UPDATE CASCADE
);

CREATE TABLE PROJECT(
utyreg varchar(20),
title varchar(100),
guide varchar(50),
period int,
core varchar(40),
CONSTRAINT FOREIGN KEY F2(utyreg) REFERENCES STU_DET(utyreg) ON DELETE
 ↪  CASCADE ON UPDATE CASCADE
);

CREATE TABLE FACULTY(
fac_name varchar(50),
fac_id varchar(20) primary key,
email varchar(50),
desig varchar(30),
joindate date,
reldate date
);

INSERT INTO STU_DET VALUES
('Reade', 1, 'KTE59XX70', '72025 Sherman Parkway', 2020, 2024, 'EEE',
 ↪  'Aswin', 'rblumer0@mapy.cz', '7044958412'),
('Gracia', 2, 'KTE96XX57', '9723 Buhler Crossing', 2020, 2024, 'CSE',
 ↪  'Amalia', 'gcrufts1@mlb.com', '6059232571'),
('Larine', 3, 'KTE99XX66', '57 Anderson Park', 2017, 2021, 'MEC', 'Arya',
 ↪  'lduffill2@histats.com', '9301870105'),
('Cyrill', 4, 'KTE05XX60', '8 Forest Circle', 2017, 2021, 'CIV',
 ↪  'Hrishikesh', 'cdemaid3@imageshack.us', '6098931285'),
('Alberto', 5, 'KTE84XX84', '65863 Stephen Pass', 2017, 2021, 'MEC',
 ↪  'Arya', 'amacmychem4@domainmarket.com', '7478791342'),
('Rowland', 6, 'KTE78XX78', '1954 Lerdahl Junction', 2017, 2021, 'CSE',
 ↪  'Shaju', 'rkimmitt5@google.co.jp', '6436096352'),
('Querida', 7, 'KTE97XX42', '75 Mayer Hill', 2017, 2021, 'CSE', 'Shaju',
 ↪  'qboyles6@dedecms.com', '7259647900'),
('Cortney', 8, 'KTE33XX46', '72058 Farragut Junction', 2017, 2021, 'EEE',
 ↪  'Abru', 'ckelbie7@simplemachines.org', '8767658796'),
('Wendell', 9, 'KTE08XX56', '5452 Clarendon Lane', 2018, 2022, 'ECE',
 ↪  'Aswin', 'wsandever8@studiopress.com', '8376102289'),
```

```
('Ruggiero', 10, 'KTE08XX40', '2050 Quincy Park', 2018, 2022, 'ECE',
↪  'Aswin', 'rdionisio9@a8.net', '9047797177'),
('Othelia', 11, 'KTE22XX47', '94542 Russell Court', 2019, 2023, 'MEC',
↪  'Hrishikesh', 'olidyarda@guardian.co.uk', '8771090635'),
('Evanne', 12, 'KTE32XX50', '7 Reindahl Avenue', 2017, 2021, 'CIV',
↪  'Hrishikesh', 'echarnickb@cdbaby.com', '8917709000'),
('Werner', 13, 'KTE67XX12', '477 Banding Point', 2017, 2021, 'MEC', 'Arya',
↪  'wfargiec@uol.com.br', '6222432090'),
('Rickie', 14, 'KTE94XX98', '4974 Clarendon Court', 2018, 2022, 'CSE',
↪  'Amalia', 'ralessandruccid@bluehost.com', '6431856826'),
('Joell', 15, 'KTE84XX57', '609 Lawn Drive', 2020, 2024, 'EEE', 'Aswin',
↪  'jrobroee@adobe.com', '6280780207'),
('Percival', 16, 'KTE52XX11', '6 Vahlen Hill', 2018, 2022, 'ECE', 'Aswin',
↪  'pcruttendenf@cmu.edu', '9673248954'),
('Kara-lynn', 17, 'KTE63XX38', '8 Delaware Park', 2018, 2022, 'CSE',
↪  'Amalia', 'kquiltyg@amazon.com', '8107687778'),
('Matty', 18, 'KTE30XX07', '173 Cherokee Court', 2020, 2024, 'CIV', 'Arya',
↪  'mmeiningerh@51.la', '7379857870'),
('Cristi', 19, 'KTE99XX47', '582 Reinke Pass', 2018, 2022, 'CSE', 'Amalia',
↪  'cscholeyi@blinklist.com', '9571826511'),
('Orel', 20, 'KTE64XX48', '54 Heath Center', 2020, 2024, 'CIV', 'Arya',
↪  'osharpj@over-blog.com', '9903333410');

INSERT INTO QUALIF VALUES
('KTE59XX70', 76, 'secondClass', 2021),
('KTE59XX70', 87, 'firstClass', 2022),
('KTE59XX70', 79, 'secondClass', 2023),
('KTE59XX70', 76, 'secondClass', 2024),
('KTE96XX57', 98, 'distinct', 2021),
('KTE96XX57', 99, 'distinct', 2022),
('KTE96XX57', 77, 'secondClass', 2023),
('KTE96XX57', 85, 'firstClass', 2024),
('KTE99XX66', 75, 'secondClass', 2018),
('KTE99XX66', 99, 'distinct', 2019),
('KTE99XX66', 82, 'firstClass', 2020),
('KTE99XX66', 94, 'distinct', 2021),
('KTE05XX60', 80, 'secondClass', 2018),
('KTE05XX60', 81, 'firstClass', 2019),
('KTE05XX60', 92, 'distinct', 2020),
('KTE05XX60', 97, 'distinct', 2021),
('KTE84XX84', 96, 'distinct', 2018),
('KTE84XX84', 77, 'secondClass', 2019),
('KTE84XX84', 97, 'distinct', 2020),
('KTE84XX84', 83, 'firstClass', 2021),
('KTE78XX78', 86, 'firstClass', 2018),
('KTE78XX78', 81, 'firstClass', 2019),
('KTE78XX78', 90, 'firstClass', 2020),
('KTE78XX78', 92, 'distinct', 2021),
('KTE97XX42', 81, 'firstClass', 2018),
```

```
('KTE97XX42', 96, 'distinct', 2019),
('KTE97XX42', 92, 'distinct', 2020),
('KTE97XX42', 92, 'distinct', 2021),
('KTE33XX46', 80, 'secondClass', 2018),
('KTE33XX46', 88, 'firstClass', 2019),
('KTE33XX46', 85, 'firstClass', 2020),
('KTE33XX46', 79, 'secondClass', 2021),
('KTE08XX56', 84, 'firstClass', 2019),
('KTE08XX56', 90, 'firstClass', 2020),
('KTE08XX56', 90, 'firstClass', 2021),
('KTE08XX56', 90, 'firstClass', 2022),
('KTE08XX40', 80, 'secondClass', 2019),
('KTE08XX40', 75, 'secondClass', 2020),
('KTE08XX40', 78, 'secondClass', 2021),
('KTE08XX40', 95, 'distinct', 2022),
('KTE22XX47', 87, 'firstClass', 2020),
('KTE22XX47', 84, 'firstClass', 2021),
('KTE22XX47', 97, 'distinct', 2022),
('KTE22XX47', 80, 'secondClass', 2023),
('KTE32XX50', 77, 'secondClass', 2018),
('KTE32XX50', 77, 'secondClass', 2019),
('KTE32XX50', 85, 'firstClass', 2020),
('KTE32XX50', 87, 'firstClass', 2021),
('KTE67XX12', 97, 'distinct', 2018),
('KTE67XX12', 99, 'distinct', 2019),
('KTE67XX12', 91, 'distinct', 2020),
('KTE67XX12', 75, 'secondClass', 2021),
('KTE94XX98', 95, 'distinct', 2019),
('KTE94XX98', 100, 'distinct', 2020),
('KTE94XX98', 84, 'firstClass', 2021),
('KTE94XX98', 100, 'distinct', 2022),
('KTE84XX57', 79, 'secondClass', 2021),
('KTE84XX57', 88, 'firstClass', 2022),
('KTE84XX57', 87, 'firstClass', 2023),
('KTE84XX57', 92, 'distinct', 2024),
('KTE52XX11', 92, 'distinct', 2019),
('KTE52XX11', 80, 'secondClass', 2020),
('KTE52XX11', 87, 'firstClass', 2021),
('KTE52XX11', 80, 'secondClass', 2022),
('KTE63XX38', 81, 'firstClass', 2019),
('KTE63XX38', 80, 'secondClass', 2020),
('KTE63XX38', 97, 'distinct', 2021),
('KTE63XX38', 80, 'secondClass', 2022),
('KTE30XX07', 94, 'distinct', 2021),
('KTE30XX07', 86, 'firstClass', 2022),
('KTE30XX07', 75, 'secondClass', 2023),
('KTE30XX07', 96, 'distinct', 2024),
('KTE99XX47', 86, 'firstClass', 2019),
('KTE99XX47', 96, 'distinct', 2020),
```

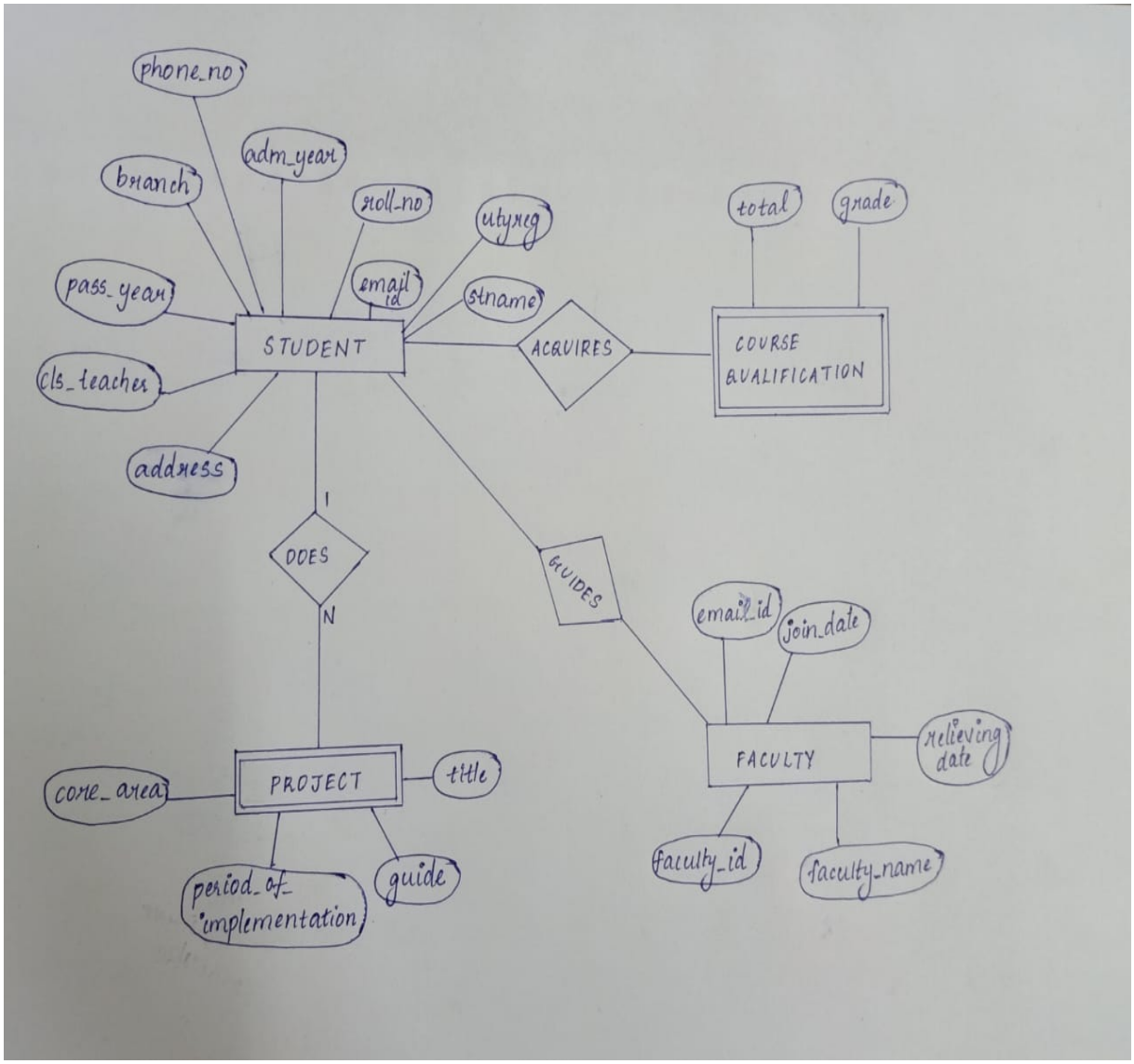
```
('KTE99XX47', 80, 'secondClass', 2021),
('KTE99XX47', 97, 'distinct', 2022),
('KTE64XX48', 96, 'distinct', 2021),
('KTE64XX48', 91, 'distinct', 2022),
('KTE64XX48', 94, 'distinct', 2023),
('KTE64XX48', 86, 'firstClass', 2024);

INSERT INTO PROJECT VALUES ('KTE59XX70', 'edu.stanford.Daltfresh', 'Aswin',
↪  2, 'ECE');
INSERT INTO PROJECT VALUES ('KTE96XX57', 'org.pbs.Toughjoyfax',
↪  'Hrishikesh', 3, 'CIV');
INSERT INTO PROJECT VALUES ('KTE99XX66', 'com.chronoengine.Matsoft',
↪  'Arya', 3, 'MEC');
INSERT INTO PROJECT VALUES ('KTE05XX60', 'org.npr.Kanlam', 'Amalia', 2,
↪  'CSE');
INSERT INTO PROJECT VALUES ('KTE84XX84', 'gov.census.Flowdesk', 'Amalia',
↪  1, 'CSE');
INSERT INTO PROJECT VALUES ('KTE78XX78', 'com.springer.Quo Lux', 'Shaju',
↪  3, 'CSE');
INSERT INTO PROJECT VALUES ('KTE97XX42', 'com.deviantart.Kanlam', 'Arya',
↪  2, 'CIV');
INSERT INTO PROJECT VALUES ('KTE33XX46', 'br.com.uol.Job', 'Aswin', 1,
↪  'ECE');
INSERT INTO PROJECT VALUES ('KTE08XX56', 'com.msn.Regrant', 'Arya', 1,
↪  'CIV');
INSERT INTO PROJECT VALUES ('KTE08XX40', 'com.friendfeed.Greenlam', 'Arya',
↪  2, 'MEC');
INSERT INTO PROJECT VALUES ('KTE22XX47', 'jp.co.yahoo.Span', 'Amalia', 1,
↪  'CSE');
INSERT INTO PROJECT VALUES ('KTE32XX50', 'com.printfriendly.Quo Lux',
↪  'Hrishikesh', 3, 'MEC');
INSERT INTO PROJECT VALUES ('KTE67XX12', 'com.blogtalkradio.Tampflex',
↪  'Arya', 3, 'CIV');
INSERT INTO PROJECT VALUES ('KTE94XX98', 'jp.co.yahoo.Daltfresh',
↪  'Hrishikesh', 1, 'CIV');
INSERT INTO PROJECT VALUES ('KTE84XX57', 'edu.wisc.Namfix', 'Aswin', 3,
↪  'ECE');
INSERT INTO PROJECT VALUES ('KTE52XX11', 'org.wikipedia.Tresom',
↪  'Hrishikesh', 2, 'CIV');
INSERT INTO PROJECT VALUES ('KTE63XX38', 'jp.co.amazon.Latlux', 'Arya', 2,
↪  'MEC');
INSERT INTO PROJECT VALUES ('KTE30XX07', 'com.forbes.Matsoft', 'Arya', 2,
↪  'CIV');
INSERT INTO PROJECT VALUES ('KTE99XX47', 'com.deviantart.Fintone', 'Aswin',
↪  1, 'EEE');
INSERT INTO PROJECT VALUES ('KTE64XX48', 'org.unicef.Kanlam', 'Abru', 3,
↪  'EEE');
```

```
INSERT INTO FACULTY VALUES ('Abru', 'KTF52YY80', 'solenane0@example.com',
↪  'Prof', '2022-05-01', '2022-10-22');
INSERT INTO FACULTY VALUES ('Arya', 'KTF45YY93', 'jliven1@posterous.com',
↪  'Prof', '2021-07-24', '2022-07-13');
INSERT INTO FACULTY VALUES ('Hrishikesh', 'KTF85YY58',
↪  'efeldmesser2@chicagotribune.com', 'Prof', '2020-09-03', null);
INSERT INTO FACULTY VALUES ('Aswin', 'KTF11YY19', 'ecantos3@walmart.com',
↪  'Ad_Hoc', '2018-01-14', null);
INSERT INTO FACULTY VALUES ('Amalia', 'KTF37YY55', 'marling4@toplist.cz',
↪  'Asst_Prof', '2021-08-30', '2022-09-23');
INSERT INTO FACULTY VALUES ('Shaju', 'KTF40YY60', 'clyle5@reddit.com',
↪  'Asst_Prof', '2018-08-17', null);

SELECT stname,stroll,addr,pass_year as Year_of_Completion,'CSE' as Branch
FROM STU_DET
WHERE branch = 'CSE'
ORDER BY pass_year;

SELECT stname,stroll,pass_year as Year_of_Completion,branch
FROM STU_DET
WHERE pass_year IS NOT NULL
ORDER BY branch,pass_year;

SELECT stname as name,stroll as rollno,P.*
FROM PROJECT P,STU_DET S
WHERE P.utyreg=S.utyreg AND branch LIKE 'CIV';

SELECT pass_year as Year_of_Project,COUNT(title) as No_of_Experiments,guide
FROM PROJECT P,STU_DET S
WHERE P.utyreg=S.utyreg
GROUP BY pass_year,guide
HAVING COUNT(title)>2;

CREATE VIEW PRE_RES AS
SELECT Q.*,branch,stname,stroll
FROM QUALIF Q,STU_DET S
WHERE S.utyreg=Q.utyreg
ORDER BY year;

CREATE VIEW TAKE AS
SELECT AVG(total) as Ac_Avg,year,branch
FROM PRE_RES
GROUP BY branch,year
ORDER BY year,Ac_Avg desc;

SELECT MAX(Ac_Avg) as Greatest_Average,year,branch
FROM TAKE
GROUP BY year
ORDER BY year;
```

```
CREATE VIEW HIGHEST AS
SELECT DISTINCT stname,stroll,S.utyreg,branch,cl_teach,e_mail,MAX(total) AS
 ↪  Highest_Score,year
FROM STU_DET S,QUALIF Q
WHERE S.utyreg=Q.utyreg
GROUP BY year,branch
ORDER BY year,branch;

SELECT * FROM HIGHEST;

SELECT stname,grade,year
FROM PRE_RES
WHERE grade LIKE 'distinct' AND year LIKE '2022';

SELECT title,guide,email,pass_year,core
FROM PROJECT P,FACULTY F,STU_DET S
WHERE guide=fac_name AND core LIKE 'CSE' AND P.utyreg=S.utyreg
ORDER BY pass_year;

SELECT adm_year,branch,COUNT(grade) AS Total_Grade
FROM STU_DET S,QUALIF Q
WHERE S.utyreg=Q.utyreg
GROUP BY year,branch
ORDER BY year,branch;

SELECT *
FROM STU_DET
WHERE adm_year=2017
ORDER BY branch;

SELECT *
FROM STU_DET
ORDER BY adm_year,branch;

SELECT branch,year,MAX(total) AS MAXIMUM,MIN(total) AS MINIMUM,AVG(total)
 ↪  AS AVERAGE
FROM PRE_RES
GROUP BY year,branch
ORDER BY year,branch;

DROP TABLE FACULTY;
DROP TABLE PROJECT;
DROP TABLE QUALIF;
DROP TABLE STU_DET;

DROP DATABASE EXPT;
```

Output

ER Diagram



```
+-----------+--------+----------------------+--------------------+--------+
| stname    | stroll | addr                 | Year_of_Completion | Branch |
+-----------+--------+----------------------+--------------------+--------+
| Rowland   |      6 | 1954 Lerdahl Junction |               2021 | CSE    |
| Querida   |      7 | 75 Mayer Hill        |               2021 | CSE    |
| Kara-lynn |     17 | 8 Delaware Park      |               2022 | CSE    |
| Rickie    |     14 | 4974 Clarendon Court |               2022 | CSE    |
| Cristi    |     19 | 582 Reinke Pass      |               2022 | CSE    |
| Gracia    |      2 | 9723 Buhler Crossing |               2024 | CSE    |
+-----------+--------+----------------------+--------------------+--------+
6 rows in set (0.00 sec)

+-----------+--------+--------------------+--------+
| stname    | stroll | Year_of_Completion | branch |
+-----------+--------+--------------------+--------+
| Cyrill    |      4 |               2021 | CIV    |
| Evanne    |     12 |               2021 | CIV    |
| Matty     |     18 |               2024 | CIV    |
| Orel      |     20 |               2024 | CIV    |
| Rowland   |      6 |               2021 | CSE    |
| Querida   |      7 |               2021 | CSE    |
| Kara-lynn |     17 |               2022 | CSE    |
| Rickie    |     14 |               2022 | CSE    |
| Cristi    |     19 |               2022 | CSE    |
| Gracia    |      2 |               2024 | CSE    |
| Ruggiero  |     10 |               2022 | ECE    |
| Wendell   |      9 |               2022 | ECE    |
| Percival  |     16 |               2022 | ECE    |
| Cortney   |      8 |               2021 | EEE    |
| Reade     |      1 |               2024 | EEE    |
| Joell     |     15 |               2024 | EEE    |
| Werner    |     13 |               2021 | MEC    |
| Alberto   |      5 |               2021 | MEC    |
| Larine    |      3 |               2021 | MEC    |
| Othelia   |     11 |               2023 | MEC    |
+-----------+--------+--------------------+--------+
20 rows in set (0.00 sec)

+--------+--------+-----------+--------------------------+------------+--------+------+
| name   | rollno | utyreg    | title                    | guide      | period | core |
+--------+--------+-----------+--------------------------+------------+--------+------+
| Cyrill |      4 | KTE05XX60 | org.npr.Kanlam           | Amalia     |      2 | CSE  |
| Matty  |     18 | KTE30XX07 | com.forbes.Matsoft       | Arya       |      2 | CIV  |
| Evanne |     12 | KTE32XX50 | com.printfriendly.Quo Lux | Hrishikesh |      3 | MEC  |
| Orel   |     20 | KTE64XX48 | org.unicef.Kanlam        | Abru       |      3 | EEE  |
+--------+--------+-----------+--------------------------+------------+--------+------+
4 rows in set (0.00 sec)
```

```
+-----------------+------------------+-------+
| Year_of_Project | No_of_Experiments | guide |
+-----------------+------------------+-------+
|            2022 |                3 | Arya  |
|            2021 |                3 | Arya  |
+-----------------+------------------+-------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

+------------------+------+--------+
| Greatest_Average | year | branch |
+------------------+------+--------+
|          89.3333 | 2018 | MEC    |
|          91.6667 | 2019 | MEC    |
|          91.6000 | 2020 | CSE    |
|          93.5000 | 2021 | CIV    |
|          97.0000 | 2022 | MEC    |
|          84.5000 | 2023 | CIV    |
|          91.0000 | 2024 | CIV    |
+------------------+------+--------+
7 rows in set (0.00 sec)
```

stname	stroll	utyreg	branch	cl_teach	e_mail	Highest_Score	year
Cyrill	4	KTE05XX60	CIV	Hrishikesh	cdemaid3@imageshack.us	80	2018
Rowland	6	KTE78XX78	CSE	Shaju	rkimmitt5@google.co.jp	86	2018
Cortney	8	KTE33XX46	EEE	Abru	ckelbie7@simplemachines.org	80	2018
Werner	13	KTE67XX12	MEC	Arya	wfargiec@uol.com.br	97	2018
Cyrill	4	KTE05XX60	CIV	Hrishikesh	cdemaid3@imageshack.us	81	2019
Kara-lynn	17	KTE63XX38	CSE	Amalia	kquiltyg@amazon.com	96	2019
Ruggiero	10	KTE08XX40	ECE	Aswin	rdionisio9@a8.net	92	2019
Cortney	8	KTE33XX46	EEE	Abru	ckelbie7@simplemachines.org	88	2019
Werner	13	KTE67XX12	MEC	Arya	wfargiec@uol.com.br	99	2019
Cyrill	4	KTE05XX60	CIV	Hrishikesh	cdemaid3@imageshack.us	92	2020
Kara-lynn	17	KTE63XX38	CSE	Amalia	kquiltyg@amazon.com	100	2020
Ruggiero	10	KTE08XX40	ECE	Aswin	rdionisio9@a8.net	90	2020
Cortney	8	KTE33XX46	EEE	Abru	ckelbie7@simplemachines.org	85	2020
Othelia	11	KTE22XX47	MEC	Hrishikesh	olidyarda@guardian.co.uk	97	2020
Cyrill	4	KTE05XX60	CIV	Hrishikesh	cdemaid3@imageshack.us	97	2021
Kara-lynn	17	KTE63XX38	CSE	Amalia	kquiltyg@amazon.com	98	2021
Ruggiero	10	KTE08XX40	ECE	Aswin	rdionisio9@a8.net	90	2021
Cortney	8	KTE33XX46	EEE	Abru	ckelbie7@simplemachines.org	79	2021
Othelia	11	KTE22XX47	MEC	Hrishikesh	olidyarda@guardian.co.uk	94	2021
Matty	18	KTE30XX07	CIV	Arya	mmeiningerh@51.la	91	2022
Kara-lynn	17	KTE63XX38	CSE	Amalia	kquiltyg@amazon.com	100	2022
Ruggiero	10	KTE08XX40	ECE	Aswin	rdionisio9@a8.net	95	2022
Reade	1	KTE59XX70	EEE	Aswin	rblumer0@mapy.cz	88	2022
Othelia	11	KTE22XX47	MEC	Hrishikesh	olidyarda@guardian.co.uk	97	2022
Matty	18	KTE30XX07	CIV	Arya	mmeiningerh@51.la	94	2023
Gracia	2	KTE96XX57	CSE	Amalia	gcrufts1@mlb.com	77	2023
Reade	1	KTE59XX70	EEE	Aswin	rblumer0@mapy.cz	87	2023
Othelia	11	KTE22XX47	MEC	Hrishikesh	olidyarda@guardian.co.uk	80	2023
Matty	18	KTE30XX07	CIV	Arya	mmeiningerh@51.la	96	2024
Gracia	2	KTE96XX57	CSE	Amalia	gcrufts1@mlb.com	85	2024
Reade	1	KTE59XX70	EEE	Aswin	rblumer0@mapy.cz	92	2024

31 rows in set (0.01 sec)

stname	grade	year
Ruggiero	distinct	2022
Othelia	distinct	2022
Orel	distinct	2022
Rickie	distinct	2022
Gracia	distinct	2022
Cristi	distinct	2022

6 rows in set (0.00 sec)

branch	year	MAXIMUM	MINIMUM	AVERAGE
CIV	2018	80	77	78.5000
CSE	2018	86	81	83.5000
EEE	2018	80	80	80.0000
MEC	2018	97	75	89.3333
CIV	2019	81	77	79.0000
CSE	2019	96	81	87.8000
ECE	2019	92	80	85.3333
EEE	2019	88	88	88.0000
MEC	2019	99	77	91.6667
CIV	2020	92	85	88.5000
CSE	2020	100	80	91.6000
ECE	2020	90	75	81.6667
EEE	2020	85	85	85.0000
MEC	2020	97	82	89.2500
CIV	2021	97	87	93.5000
CSE	2021	98	80	90.5000
ECE	2021	90	78	85.0000
EEE	2021	79	76	78.0000
MEC	2021	94	75	84.0000
CIV	2022	91	86	88.5000
CSE	2022	100	80	94.0000
ECE	2022	95	80	88.3333
EEE	2022	88	87	87.5000
MEC	2022	97	97	97.0000
CIV	2023	94	75	84.5000
CSE	2023	77	77	77.0000
EEE	2023	87	79	83.0000
MEC	2023	80	80	80.0000
CIV	2024	96	86	91.0000
CSE	2024	85	85	85.0000
EEE	2024	92	76	84.0000

31 rows in set (0.00 sec)

```
+----------------------+--------+--------------------+-----------+------+
| title                | guide  | email              | pass_year | core |
+----------------------+--------+--------------------+-----------+------+
| gov.census.Flowdesk  | Amalia | marling4@toplist.cz |      2021 | CSE  |
| org.npr.Kanlam       | Amalia | marling4@toplist.cz |      2021 | CSE  |
| com.springer.Quo Lux | Shaju  | clyle5@reddit.com    |      2021 | CSE  |
| jp.co.yahoo.Span     | Amalia | marling4@toplist.cz |      2023 | CSE  |
+----------------------+--------+--------------------+-----------+------+
4 rows in set (0.01 sec)

+----------+--------+-------------+
| adm_year | branch | Total_Grade |
+----------+--------+-------------+
|     2017 | CIV    |           2 |
|     2017 | CSE    |           2 |
|     2017 | EEE    |           1 |
|     2017 | MEC    |           3 |
|     2017 | CIV    |           2 |
|     2018 | CSE    |           5 |
|     2018 | ECE    |           3 |
|     2017 | EEE    |           1 |
|     2017 | MEC    |           3 |
|     2017 | CIV    |           2 |
|     2018 | CSE    |           5 |
|     2018 | ECE    |           3 |
|     2017 | EEE    |           1 |
|     2019 | MEC    |           4 |
|     2017 | CIV    |           4 |
|     2018 | CSE    |           6 |
|     2018 | ECE    |           3 |
|     2017 | EEE    |           3 |
|     2019 | MEC    |           4 |
|     2020 | CIV    |           2 |
|     2018 | CSE    |           4 |
|     2018 | ECE    |           3 |
|     2020 | EEE    |           2 |
|     2019 | MEC    |           1 |
|     2020 | CIV    |           2 |
|     2020 | CSE    |           1 |
|     2020 | EEE    |           2 |
|     2019 | MEC    |           1 |
|     2020 | CIV    |           2 |
|     2020 | CSE    |           1 |
|     2020 | EEE    |           2 |
+----------+--------+-------------+
31 rows in set (0.00 sec)
```

stname	stroll	utyreg	addr	adm_year	pass_year	branch	cl_teach	e_mail	phno
Cyrill	4	KTE05XX60	8 Forest Circle	2017	2021	CIV	Hrishikesh	cdemaid3@imageshack.us	6098931285
Evanne	12	KTE32XX50	7 Reindahl Avenue	2017	2021	CIV	Hrishikesh	echarnickb@cdbaby.com	8917709000
Rowland	6	KTE78XX78	1954 Lerdahl Junction	2017	2021	CSE	Shaju	rkimmitt5@google.co.jp	6436096352
Querida	7	KTE97XX42	75 Mayer Hill	2017	2021	CSE	Shaju	qboyles6@dedecms.com	7259647900
Cortney	8	KTE33XX46	72058 Farragut Junction	2017	2021	EEE	Abru	ckelbie7@simplemachines.org	8767658796
Werner	13	KTE67XX12	477 Banding Point	2017	2021	MEC	Arya	wfargiec@uol.com.br	6222432090
Alberto	5	KTE84XX84	65863 Stephen Pass	2017	2021	MEC	Arya	amacmychem4@domainmarket.com	7478791342
Larine	3	KTE99XX66	57 Anderson Park	2017	2021	MEC	Arya	lduffill2@histats.com	9301870105

8 rows in set (0.00 sec)

stname	stroll	utyreg	addr	adm_year	pass_year	branch	cl_teach	e_mail	phno
Cyrill	4	KTE05XX60	8 Forest Circle	2017	2021	CIV	Hrishikesh	cdemaid3@imageshack.us	6098931285
Evanne	12	KTE32XX50	7 Reindahl Avenue	2017	2021	CIV	Hrishikesh	echarnickb@cdbaby.com	8917709000
Rowland	6	KTE78XX78	1954 Lerdahl Junction	2017	2021	CSE	Shaju	rkimmitt5@google.co.jp	6436096352
Querida	7	KTE97XX42	75 Mayer Hill	2017	2021	CSE	Shaju	qboyles6@dedecms.com	7259647900
Cortney	8	KTE33XX46	72058 Farragut Junction	2017	2021	EEE	Abru	ckelbie7@simplemachines.org	8767658796
Werner	13	KTE67XX12	477 Banding Point	2017	2021	MEC	Arya	wfargiec@uol.com.br	6222432090
Alberto	5	KTE84XX84	65863 Stephen Pass	2017	2021	MEC	Arya	amacmychem4@domainmarket.com	7478791342
Larine	3	KTE99XX66	57 Anderson Park	2017	2021	MEC	Arya	lduffill2@histats.com	9301870105
Kara-lynn	17	KTE63XX38	8 Delaware Park	2018	2022	CSE	Amalia	kquiltyg@amazon.com	8107687778
Rickie	14	KTE94XX98	4974 Clarendon Court	2018	2022	CSE	Amalia	ralessandruccid@bluehost.com	6431856826
Cristi	19	KTE99XX47	582 Reinke Pass	2018	2022	CSE	Amalia	cscholeyi@blinklist.com	9571826511
Ruggiero	10	KTE08XX40	2050 Quincy Park	2018	2022	ECE	Aswin	rdionisio9@a8.net	9047797177
Wendell	9	KTE08XX56	5452 Clarendon Lane	2018	2022	ECE	Aswin	wsandever8@studiopress.com	8376102289
Percival	16	KTE52XX11	6 Vahlen Hill	2018	2022	ECE	Aswin	pcruttendenf@cmu.edu	9673248954
Othelia	11	KTE22XX47	94542 Russell Court	2019	2023	MEC	Hrishikesh	olidyarda@guardian.co.uk	8771090635
Matty	18	KTE30XX07	173 Cherokee Court	2020	2024	CIV	Arya	mmeiningerh@51.la	7379857870
Orel	20	KTE64XX48	54 Heath Center	2020	2024	CIV	Arya	osharpj@over-blog.com	9903333410
Gracia	2	KTE96XX57	9723 Buhler Crossing	2020	2024	CSE	Amalia	gcrufts1@mlb.com	6059232571
Reade	1	KTE59XX70	72025 Sherman Parkway	2020	2024	EEE	Aswin	rblumer0@mapy.cz	7044958412
Joell	15	KTE84XX57	609 Lawn Drive	2020	2024	EEE	Aswin	jrobroee@adobe.com	6280780207

20 rows in set (0.00 sec)

Figure 7: Output of Experiment 7

Experiment 8

Even or Odd
Date:-05/02/2023

Question:- Write a PL/SQL code to check whether a number is even or odd.

Answer:-

The SQL script for solving the given question is the following.

```
DECLARE
        n NUMBER := &n;
BEGIN
        IF( mod(n,2)=0) THEN
                dbms_output.put_line('Even Number');
        ELSE
                dbms_output.put_line('Odd Number');
        END IF;
END;
/
```

Output

```
SQL> set serveroutput on
SQL> @C:\DBMS_Lab\PLSQL_1.sql
Enter value for n: 4
old   2:        n NUMBER := &n;
new   2:        n NUMBER := 4;
Even Number

PL/SQL procedure successfully completed.

SQL> @C:\DBMS_Lab\PLSQL_1.sql
Enter value for n: 7
old   2:        n NUMBER := &n;
new   2:        n NUMBER := 7;
Odd Number

PL/SQL procedure successfully completed.
```

Figure 8: Output of Experiment 8

Experiment 9

Prime or Composite
Date:-05/02/2023

Question:- Write a PL/SQL code to check whether a number is prime or not.

Answer:-

The SQL script for solving the given question is the following.

```
DECLARE
        n NUMBER := &n;
        i NUMBER := 2;
        flag NUMBER := 1;
BEGIN
        FOR i in 2..n/2
        LOOP
                IF( mod(n,i)=0 ) THEN
                        flag := 0;
                        exit;
                END IF;
        END LOOP;
        IF( flag=1 ) THEN
                dbms_output.put_line('Prime Number');
        ELSE
                dbms_output.put_line('Composite Number');
        END IF;
END;
/
```

Output

```
SQL> set serveroutput on
SQL> @C:\DBMS_Lab\PLSQL_2.sql
Enter value for n: 6
old   2:        n NUMBER := &n;
new   2:        n NUMBER := 6;
Composite Number

PL/SQL procedure successfully completed.

SQL> @C:\DBMS_Lab\PLSQL_2.sql
Enter value for n: 2
old   2:        n NUMBER := &n;
new   2:        n NUMBER := 2;
Prime Number

PL/SQL procedure successfully completed.
```

Figure 9: Output of Experiment 9

chapter*Experiment 10

Factorial
Date:-05/02/2023

Question:- Write a PL/SQL code to find the factorial of a number.

Answer:-

The SQL script for solving the given question is the following.

```
DECLARE
        n NUMBER := &n;
        f NUMBER := 1;
BEGIN
        FOR i in 1..n
        LOOP
                f := f*i;
        END LOOP;
        dbms_output.put_line('Factorial is '|| f);
END;
/
```

Output

```
SQL> set serveroutput on
SQL> @C:\DBMS_Lab\PLSQL_3.sql
Enter value for n: 4
old   2:        n NUMBER := &n;
new   2:        n NUMBER := 4;
Factorial is 24

PL/SQL procedure successfully completed.

SQL> @C:\DBMS_Lab\PLSQL_3.sql
Enter value for n: 9
old   2:        n NUMBER := &n;
new   2:        n NUMBER := 9;
Factorial is 362880

PL/SQL procedure successfully completed.
```

Figure 10: Output of Experiment 10

Experiment 11

Perfect Number
Date:-05/02/2023

Question:- Write a PL/SQL code to check whether a number is perfect or not.

Answer:-

The SQL script for solving the given question is the following.

```
DECLARE
        n NUMBER := &n;
        i NUMBER;
        t NUMBER := 0;
BEGIN
        FOR i in 1..n/2
        LOOP
                IF( n mod i = 0) THEN
                        t := t+i;
                END IF;
        END LOOP;
        IF( n=t ) THEN
                dbms_output.put_line('Perfect number');
        ELSE
                dbms_output.put_line('Not a Perfect number');
        END IF;
END;
/
```

Output

```
SQL> @C:\DBMS_Lab\PLSQL_4.sql
Enter value for n: 6
old   2:       n NUMBER := &n;
new   2:       n NUMBER := 6;
Perfect number

PL/SQL procedure successfully completed.

SQL> @C:\DBMS_Lab\PLSQL_4.sql
Enter value for n: 25
old   2:       n NUMBER := &n;
new   2:       n NUMBER := 25;
Not a Perfect number

PL/SQL procedure successfully completed.
```

Figure 11: Output of Experiment 11

Experiment 12

Calculator
Date:-05/02/2023

Question:- Write a PL/SQL code to create a calculator.

Answer:-

The SQL script for solving the given question is the following.

```
DECLARE
        n NUMBER := &n;
        m NUMBER := &m;
        op VARCHAR2(20) := '&op';
        rs number;
BEGIN
        IF( op = '+') THEN
                rs := n+m;
        ELSIF( op = '-') THEN
                rs := n-m;
        ELSIF( op = '*') THEN
                rs := n*m;
        ELSIF( op = '/') THEN
                rs := n/m;
        ELSE
                dbms_output.put_line('Error...');
        END IF;
        dbms_output.put_line(rs);
END;
/
```

Output

```
SQL> @C:\DBMS_Lab\PLSQL_5.sql
Enter value for n: 9
old   2:        n NUMBER := &n;
new   2:        n NUMBER := 9;
Enter value for m: 3
old   3:        m NUMBER := &m;
new   3:        m NUMBER := 3;
Enter value for op: /
old   4:        op VARCHAR2(20) := '&op';
new   4:        op VARCHAR2(20) := '/';
3

PL/SQL procedure successfully completed.
```

```
SQL> set serveroutput on
SQL> @C:\DBMS_Lab\PLSQL_5.sql
Enter value for n: 7
old   2:        n NUMBER := &n;
new   2:        n NUMBER := 7;
Enter value for m: 3
old   3:        m NUMBER := &m;
new   3:        m NUMBER := 3;
Enter value for op: +
old   4:        op VARCHAR2(20) := '&op';
new   4:        op VARCHAR2(20) := '+';
10

PL/SQL procedure successfully completed.

SQL> @C:\DBMS_Lab\PLSQL_5.sql
Enter value for n: 6
old   2:        n NUMBER := &n;
new   2:        n NUMBER := 6;
Enter value for m: 2
old   3:        m NUMBER := &m;
new   3:        m NUMBER := 2;
Enter value for op: -
old   4:        op VARCHAR2(20) := '&op';
new   4:        op VARCHAR2(20) := '-';
4

PL/SQL procedure successfully completed.

SQL> @C:\DBMS_Lab\PLSQL_5.sql
Enter value for n: 7
old   2:        n NUMBER := &n;
new   2:        n NUMBER := 7;
Enter value for m: 4
old   3:        m NUMBER := &m;
new   3:        m NUMBER := 4;
Enter value for op: *
old   4:        op VARCHAR2(20) := '&op';
new   4:        op VARCHAR2(20) := '*';
28

PL/SQL procedure successfully completed.
```

Figure 12: Output of Experiment 12

Experiment 13

Age Calculation
Date:-06/02/2023

Question:- Given the scheme Person (pid, pname, DOB) . Find the age of each person using cursor.

Answer:-

The SQL script for solving the given question is the following.

```
CREATE TABLE PERSON(
PID int,
P_name varchar(15),
DOB date
);

INSERT INTO PERSON(PID,P_name,DOB) VALUES (1,'Aswathy','26-05-2002') ;
INSERT INTO PERSON(PID,P_name,DOB) VALUES (2,'Amalu','16-09-2000') ;
INSERT INTO PERSON(PID,P_name,DOB) VALUES (3,'Jerry','10-10-1999') ;
INSERT INTO PERSON(PID,P_name,DOB) VALUES (4,'Nandhu','06-09-2001');
INSERT INTO PERSON(PID,P_name,DOB) VALUES (5,'Abhi','17-03-2001') ;
INSERT INTO PERSON(PID,P_name,DOB) VALUES (6,'Ansa','22-05-2001') ;
INSERT INTO PERSON(PID,P_name,DOB) VALUES (7,'Julie','03-10-2002') ;
INSERT INTO PERSON(PID,P_name,DOB) VALUES (8,'Arya','03-06-2002');
INSERT INTO PERSON(PID,P_name,DOB) VALUES (9,'Manu','29-01-2001') ;
INSERT INTO PERSON(PID,P_name,DOB) VALUES (10,'Annie','21-07-2001') ;
SELECT * FROM PERSON;

/

DECLARE
    name_ PERSON.P_name%TYPE;
    age int;
    CURSOR Age IS
    SELECT P_name,MONTHS_BETWEEN(sysdate , DOB) / 12 age FROM PERSON;
BEGIN
    OPEN AgeCalculation;
    LOOP
        FETCH Age INTO name_,age;
        EXIT WHEN Age%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(name_ || ' is ' || age || ' years old');
    END LOOP;
    CLOSE Age;
END;
/
```

Output

```
       PID PNAME                DOB
---------- -------------------- ----------
         1 Aswathy              26-05-2002
         2 Amalu                16-09-2000
         3 Jerry                10-10-1999
         4 Nandhu               06-09-2001
         5 Abhi                 17-03-2001
         6 Ansa                 22-05-2001
         7 Julie                03-10-2002
         8 Arya                 03-06-2002
         9 Manu                 29-01-2001
        10 Annie                21-07-2001

10 rows selected.


       PID PNAME                DOB
---------- -------------------- ----------
         1 Aswathy              26-05-2002
         2 Amalu                16-09-2000
         3 Jerry                10-10-1999
         4 Nandhu               06-09-2001
         5 Abhi                 17-03-2001
         6 Ansa                 22-05-2001
         7 Julie                03-10-2002
         8 Arya                 03-06-2002
         9 Manu                 29-01-2001
        10 Annie                21-07-2001

10 rows selected.

Aswathy is 21 years old
Amalu is 22 years old
Jerry is 23 years old
Nandhu is 21 years old
Abhi is 22 years old
Ansa is 22 years old
Julie is 20 years old
Arya is 21 years old
Manu is 22 years old
Annie is 22 years old

PL/SQL procedure successfully completed.
```

Figure 13: Output of Experiment 13

Experiment 14

Employee Database
Date:-06/02/2023

Question:- Given the schema Employee (empid, empname, joining date, relieving date, salary)
(a) Find the service (in years) for each relieved employee.
(b) Find the Pension amount to be paid to each relieved employee. (Pension is equal to the years of service *salary divided by 100.)
Use cursors.

Answer:-
The SQL script for solving the given question is the following.

```
CREATE TABLE EMPLOYEE(
empid varchar2(20),
empname varchar2(40),
joining_date date,
relieving_date date,
salary real
);

insert into Employee (empid, empname, joining_date, relieving_date, salary)
 ↪  values ('51-5841708', 'See Routhorn', '14-05-2007', '05-05-2019',
 ↪  29121.47);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
 ↪  values ('85-2684047', 'Elisabet Celli', '21-10-1987', '09-01-2021',
 ↪  12733.43);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
 ↪  values ('64-6740456', 'Sari Magovern', '10-10-1995', '07-12-2019',
 ↪  196485.67);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
 ↪  values ('93-7185619', 'Stephannie O''Cosgra', '30-01-1982',
 ↪  '10-07-2015', 93232.95);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
 ↪  values ('03-5201744', 'Annmarie Wisham', '20-10-1991', '24-07-2016',
 ↪  164716.84);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
 ↪  values ('33-5111972', 'Leland Bamber', '30-08-2000', '27-12-2016',
 ↪  86504.36);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
 ↪  values ('95-0417404', 'May Screwton', '09-02-1984', '22-12-2017',
 ↪  186857.34);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
 ↪  values ('79-1519824', 'Arlee Gotfrey', '21-01-1986', '16-02-2016',
 ↪  14557.12);
```

```
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('54-9079903', 'Dehlia MacGuffog', '21-02-1999', '22-06-2021',
↪  40649.83);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('55-5070347', 'Melvin MacGown', '24-07-1985', '13-06-2018',
↪  188499.16);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('22-4204327', 'Chance Sneaker', '16-06-2006', '24-08-2015',
↪  23758.62);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('71-8399513', 'Kevan Lashbrook', '19-10-1992', '21-01-2018',
↪  84778.93);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('71-7234947', 'Blondie Anwyl', '17-07-2003', '24-05-2016',
↪  163845.56);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('99-2294688', 'Oran Brusle', '31-10-1984', '18-05-2017',
↪  55639.65);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('81-1200905', 'Archy Vereker', '26-06-1991', '19-11-2019',
↪  120202.55);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('45-8756534', 'Donielle Berardt', '31-12-1998', '14-04-2015',
↪  138526.63);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('95-6085475', 'Roderick Corps', '14-11-1982', '24-04-2022',
↪  40173.60);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('13-2876536', 'Marylou Pucknell', '01-01-1998', '04-05-2019',
↪  127381.42);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('17-0781401', 'Olvan Laugharne', '06-04-2008', '03-01-2022',
↪  38120.12);
insert into Employee (empid, empname, joining_date, relieving_date, salary)
↪  values ('52-7162620', 'Orsola Angier', '10-01-1987', '13-11-2016',
↪  114136.65);

SELECT * FROM EMPLOYEE;

/

DECLARE
    id EMPLOYEE.empid%TYPE;
    name_ EMPLOYEE.empname%TYPE;
    serv number(2,0);
    pension number(10,2);
    sal EMPLOYEE.salary%TYPE;
    CURSOR Details IS
```

```
    SELECT empid,empname,MONTHS_BETWEEN(relieving_date,joining_date)/12
     ↪  yea,salary
    FROM EMPLOYEE;
BEGIN
    OPEN Details;
    LOOP
        FETCH Details INTO id,name_,serv,sal;
        EXIT WHEN Details%NOTFOUND;
        pension := serv*sal/100;
        DBMS_OUTPUT.PUT_LINE('ID: ' || id || ' Name: ' || name_ || ' Years
         ↪  of Service: ' || serv || ' Pension: ' || pension);
    END LOOP;
    CLOSE Details;
END;
/
```

Output

C:\Users\aswat\Downloads\WINDOWS.X64_213000_db_home\bin\sqlplus.exe

```
------------------- ---------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
51-5841708          See Routhorn                             14-05-2007
05-05-2019   29121.47

85-2684047          Elisabet Celli                           21-10-1987
09-01-2021   12733.43

64-6740456          Sari Magovern                            10-10-1995
07-12-2019  196485.67


EMPID               EMPNAME                                  JOINING_DA
------------------- ---------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
93-7185619          Stephannie O'Cosgra                      30-01-1982
10-07-2015   93232.95

03-5201744          Annmarie Wisham                          20-10-1991
24-07-2016  164716.84

33-5111972          Leland Bamber                            30-08-2000
27-12-2016   86504.36


EMPID               EMPNAME                                  JOINING_DA
------------------- ---------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
95-0417404          May Screwton                             09-02-1984
22-12-2017  186857.34

79-1519824          Arlee Gotfrey                            21-01-1986
16-02-2016   14557.12

54-9079903          Dehlia MacGuffog                         21-02-1999
22-06-2021   40649.83


EMPID               EMPNAME                                  JOINING_DA
------------------- ---------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
55-5070347          Melvin MacGown                           24-07-1985
13-06-2018  188499.16

22-4204327          Chance Sneaker                           16-06-2006
```

```
71-8399513          Kevan Lashbrook                         19-10-1992
21-01-2018   84778.93


EMPID               EMPNAME                                 JOINING_DA
------------------- --------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
71-7234947          Blondie Anwyl                           17-07-2003
24-05-2016  163845.56

99-2294688          Oran Brusle                             31-10-1984
18-05-2017   55639.65

81-1200905          Archy Vereker                           26-06-1991
19-11-2019  120202.55


EMPID               EMPNAME                                 JOINING_DA
------------------- --------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
45-8756534          Donielle Berardt                        31-12-1998
14-04-2015  138526.63

95-6085475          Roderick Corps                          14-11-1982
24-04-2022    40173.6

13-2876536          Marylou Pucknell                        01-01-1998
04-05-2019  127381.42


EMPID               EMPNAME                                 JOINING_DA
------------------- --------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
17-0781401          Olvan Laugharne                         06-04-2008
03-01-2022   38120.12

52-7162620          Orsola Angier                           10-01-1987
13-11-2016  114136.65


20 rows selected.
```

```
EMPID               EMPNAME                                  JOINING_DA
------------------- ---------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
51-5841708          See Routhorn                             14-05-2007
05-05-2019   29121.47

85-2684047          Elisabet Celli                           21-10-1987
09-01-2021   12733.43

64-6740456          Sari Magovern                            10-10-1995
07-12-2019  196485.67


EMPID               EMPNAME                                  JOINING_DA
------------------- ---------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
93-7185619          Stephannie O'Cosgra                      30-01-1982
10-07-2015   93232.95

03-5201744          Annmarie Wisham                          20-10-1991
24-07-2016  164716.84

33-5111972          Leland Bamber                            30-08-2000
27-12-2016   86504.36


EMPID               EMPNAME                                  JOINING_DA
------------------- ---------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
95-0417404          May Screwton                             09-02-1984
22-12-2017  186857.34

79-1519824          Arlee Gotfrey                            21-01-1986
16-02-2016   14557.12

54-9079903          Dehlia MacGuffog                         21-02-1999
22-06-2021   40649.83


EMPID               EMPNAME                                  JOINING_DA
------------------- ---------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
55-5070347          Melvin MacGown                           24-07-1985
13-06-2018  188499.16
```

```
EMPID               EMPNAME                                JOINING_DA
------------------- -------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
71-7234947          Blondie Anwyl                          17-07-2003
24-05-2016  163845.56

99-2294688          Oran Brusle                            31-10-1984
18-05-2017   55639.65

81-1200905          Archy Vereker                          26-06-1991
19-11-2019  120202.55


EMPID               EMPNAME                                JOINING_DA
------------------- -------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
45-8756534          Donielle Berardt                       31-12-1998
14-04-2015  138526.63

95-6085475          Roderick Corps                         14-11-1982
24-04-2022    40173.6

13-2876536          Marylou Pucknell                       01-01-1998
04-05-2019  127381.42


EMPID               EMPNAME                                JOINING_DA
------------------- -------------------------------------- ----------
RELIEVING_     SALARY
---------- ----------
17-0781401          Olvan Laugharne                        06-04-2008
03-01-2022   38120.12

52-7162620          Orsola Angier                          10-01-1987
13-11-2016  114136.65
```

```
ID: 51-5841708 Name: See Routhorn Years of Service: 12 Pension: 3494.58
ID: 85-2684047 Name: Elisabet Celli Years of Service: 33 Pension: 4202.03
ID: 64-6740456 Name: Sari Magovern Years of Service: 24 Pension: 47156.56
ID: 93-7185619 Name: Stephannie O'Cosgra Years of Service: 33 Pension: 30766.87
ID: 03-5201744 Name: Annmarie Wisham Years of Service: 25 Pension: 41179.21
ID: 33-5111972 Name: Leland Bamber Years of Service: 16 Pension: 13840.7
ID: 95-0417404 Name: May Screwton Years of Service: 34 Pension: 63531.5
ID: 79-1519824 Name: Arlee Gotfrey Years of Service: 30 Pension: 4367.14
ID: 54-9079903 Name: Dehlia MacGuffog Years of Service: 22 Pension: 8942.96
ID: 55-5070347 Name: Melvin MacGown Years of Service: 33 Pension: 62204.72
ID: 22-4204327 Name: Chance Sneaker Years of Service: 9 Pension: 2138.28
ID: 71-8399513 Name: Kevan Lashbrook Years of Service: 25 Pension: 21194.73
ID: 71-7234947 Name: Blondie Anwyl Years of Service: 13 Pension: 21299.92
ID: 99-2294688 Name: Oran Brusle Years of Service: 33 Pension: 18361.08
ID: 81-1200905 Name: Archy Vereker Years of Service: 28 Pension: 33656.71
ID: 45-8756534 Name: Donielle Berardt Years of Service: 16 Pension: 22164.26
ID: 95-6085475 Name: Roderick Corps Years of Service: 39 Pension: 15667.7
ID: 13-2876536 Name: Marylou Pucknell Years of Service: 21 Pension: 26750.1
ID: 17-0781401 Name: Olvan Laugharne Years of Service: 14 Pension: 5336.82
ID: 52-7162620 Name: Orsola Angier Years of Service: 30 Pension: 34241

PL/SQL procedure successfully completed.
```

Figure 14: Output of Experiment 14

Experiment 15

Rank Determination
Date:-06/02/2023

Question:- Write a pl/sql code to insert several names, roll nos and marks of three subjects for the students of a class into a table named student and compute their rank list and insert the rank information into the same table.

Answer:-

The SQL script for solving the given question is the following.

```
CREATE TABLE STUDENT(
roll int primary key,
name_ varchar2(30),
sub1 int,
sub2 int,
sub3 int,
total_ int,
rank int
);

set serveroutput on

DECLARE
    CURSOR DETAILS IS
    SELECT roll , name_ , total_ , rank() over (order by total_ desc) as
     ↪  rank
    FROM STUDENT;
    stud DETAILS%rowtype;
BEGIN
    insert into STUDENT (roll, name_, sub1, sub2, sub3,total_) values (1,
     ↪  'Melodee Corbishley', 28, 81, 16,125);
    insert into STUDENT (roll, name_, sub1, sub2, sub3,total_) values (2,
     ↪  'Mike Thurner', 86, 36, 54,176);
    insert into STUDENT (roll, name_, sub1, sub2, sub3,total_) values (3,
     ↪  'Nydia Tonry', 32, 80, 71,183);
    insert into STUDENT (roll, name_, sub1, sub2, sub3,total_) values (4,
     ↪  'Pearle Piddock', 16, 54, 31,101);
    insert into STUDENT (roll, name_, sub1, sub2, sub3,total_) values (5,
     ↪  'Lennard Fishly', 39, 100, 18,157);
    insert into STUDENT (roll, name_, sub1, sub2, sub3,total_) values (6,
     ↪  'Rosita Ionesco', 79, 91, 16,186);
    insert into STUDENT (roll, name_, sub1, sub2, sub3,total_) values (7,
     ↪  'Terrie Denness', 81, 59, 21,161);
    insert into STUDENT (roll, name_, sub1, sub2, sub3,total_) values (8,
     ↪  'Prentiss Leale', 4, 32, 18,54);
    insert into STUDENT (roll, name_, sub1, sub2, sub3,total_) values (9,
     ↪  'Torrey Iacopo', 44, 92, 81,217);
```

```
    insert into STUDENT (roll, name_, sub1, sub2, sub3,total_) values (10,
     ↪  'Wrennie Crotch', 84, 98, 66,248);

    OPEN DETAILS;
    LOOP
        FETCH DETAILS INTO stud;
        EXIT WHEN DETAILS%NOTFOUND;
        UPDATE STUDENT SET rank = stud.rank where roll=stud.roll;
    END LOOP;
    CLOSE DETAILS;
END;
/
SELECT * FROM STUDENT ORDER BY student.rank;
```

Output

```
     ROLL NAME_                          SUB1       SUB2       SUB3
--------- ------------------------------ ---------- ---------- ----------
   TOTAL_       RANK
--------- ----------
       10 Wrennie Crotch                      84         98         66
      248          1

        9 Torrey Iacopo                       44         92         81
      217          2

        6 Rosita Ionesco                      79         91         16
      186          3


     ROLL NAME_                          SUB1       SUB2       SUB3
--------- ------------------------------ ---------- ---------- ----------
   TOTAL_       RANK
--------- ----------
        3 Nydia Tonry                         32         80         71
      183          4

        2 Mike Thurner                        86         36         54
      176          5

        7 Terrie Denness                      81         59         21
      161          6


     ROLL NAME_                          SUB1       SUB2       SUB3
--------- ------------------------------ ---------- ---------- ----------
   TOTAL_       RANK
--------- ----------
        5 Lennard Fishly                      39        100         18
      157          7

        1 Melodee Corbishley                  28         81         16
      125          8

        4 Pearle Piddock                      16         54         31
      101          9


     ROLL NAME_                          SUB1       SUB2       SUB3
--------- ------------------------------ ---------- ---------- ----------
   TOTAL_       RANK
--------- ----------
        8 Prentiss Leale                       4         32         18
       54         10
```

Figure 15: Output of Experiment 15

Experiment 16

Employee DB
Date:-06/02/2023

Question:- The following table shows the salary information of employees in a company. EMPLOYEE (empid, empname, designation, dept, salary) Write a trigger that displays the total number of tuples in the relation on each insertion, deletion and updation.

Answer:-

The SQL script for solving the given question is the following.

```
CREATE TABLE Employee (
empid int PRIMARY KEY,
empname VARCHAR(50),
designation VARCHAR(50),
dept VARCHAR(50),
salary int
);
set serveroutput on
CREATE OR REPLACE TRIGGER employee_trig
BEFORE INSERT OR DELETE OR UPDATE ON Employee
FOR EACH ROW
DECLARE
v_count NUMBER;
BEGIN
SELECT COUNT(*) INTO v_count FROM Employee;
DBMS_OUTPUT.PUT_LINE('Number of tuples in Employee relation: ' ||,→
 ↪  v_count);
END;
/
INSERT INTO Employee VALUES(22,'Aswathy S','Team Lead','AI',10000);
```

Output

```
Number of tuples in Employee relation: 1

1 row(s) inserted.

0.01 seconds
```

Figure 16: Output of Experiment 16

Experiment 17

Salary Database
Date:-06/02/2023

Question:- The following table shows the salary information of employees in a company. EMPLOYEE (empid, empname, salary) Write a trigger that causes insertion of a new entry into the table INCREMENT(empid, incr), if the difference arising due to an updation of the salary of an existing employee is greater than Rs. 1000/-.

Answer:-

The SQL script for solving the given question is the following.

```
CREATE TABLE Employees
(empid varchar(10) PRIMARY KEY,
empname VARCHAR(30),
salary int);
INSERT INTO Employees VALUES('71512','Joseph',75000);
INSERT INTO Employees VALUES('62112','George',56000);
INSERT INTO Employees VALUES('51912','Malavika',33000);
INSERT INTO Employees VALUES('81512','Rameh',80000);
INSERT INTO Employees VALUES('11512','Kailas',55000);

CREATE TABLE Increments (
  empid varchar(10) PRIMARY KEY,
  incr int,
  FOREIGN KEY(empid) references Employees(empid)
);
--creating trigger

CREATE OR REPLACE TRIGGER Increments
AFTER UPDATE OF salary ON Employees
FOR EACH ROW
DECLARE
  diff NUMBER;
BEGIN
  diff := :NEW.salary - :OLD.salary;
  IF diff > 1000 THEN
    INSERT INTO Increments (empid, incr)
    VALUES (:NEW.empid, diff);
  END IF;
END;
/

--updating salary of an employee

update Employees
set salary = 35000
where empid=51912;
```

```
select * from Increments;
```

Output

Figure 17: Output of Experiment 17

Project Report

Student Database Management System

Description

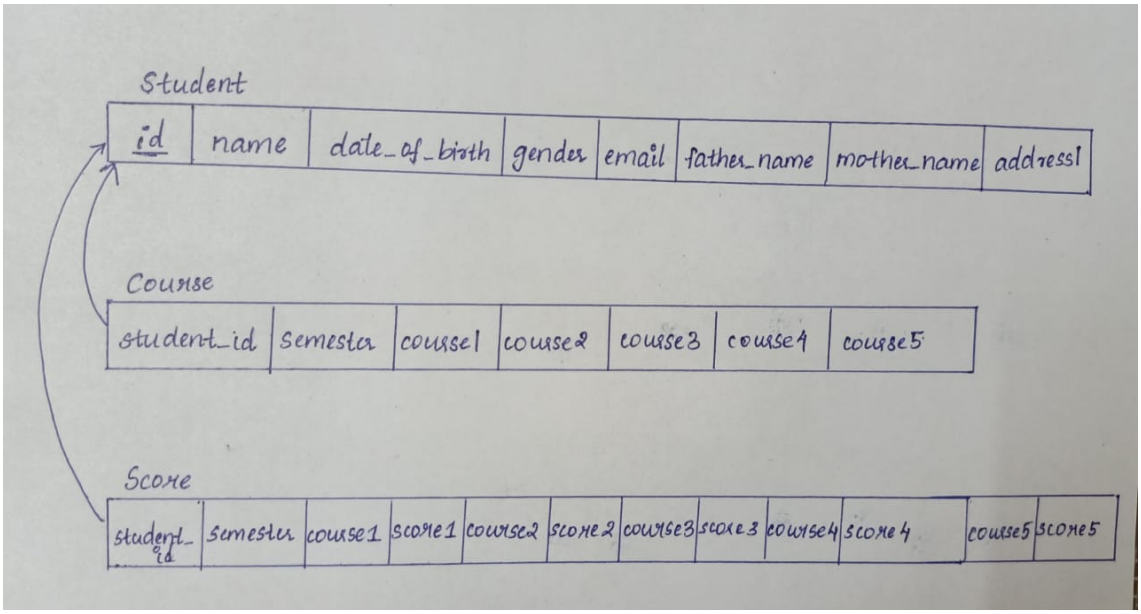
The Student Database Management System is developed for the use in educational institutions, primarily for the faculty.

The system can be used for keeping the records of the students.Only faculties have access to this system.Various functionalities are provided by the system which include entering the personal information of a new student, entering the details of the courses selected by the student in each semester and the corresponding courses. Also the mark list of a student can be obtained in a PDF format.Updation of details entered are also possible in this system.

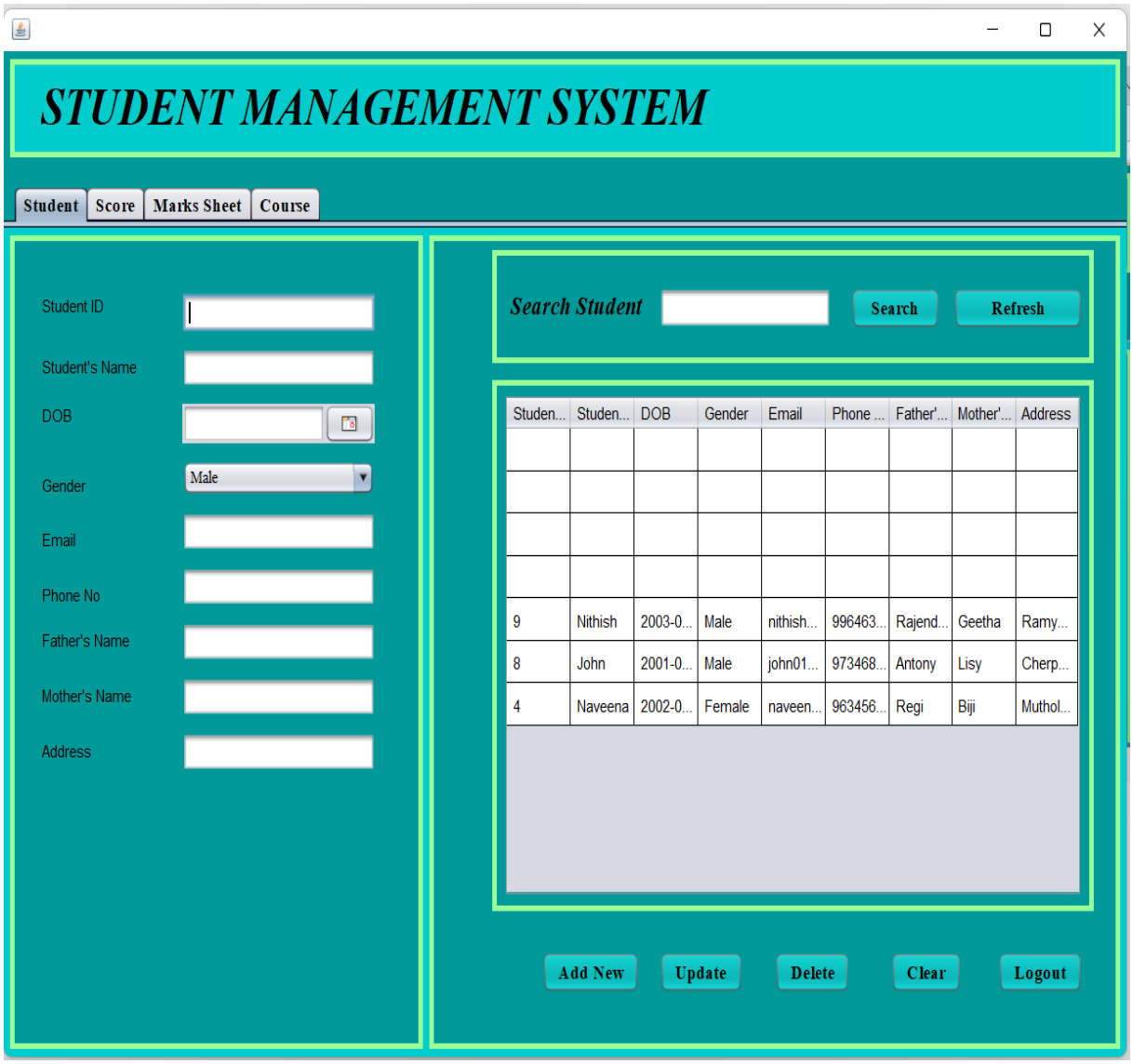
The details about the student including the courses selected and the score can be searched using the student's ID.The details of students who are no longer a part of the institution can be deleted.

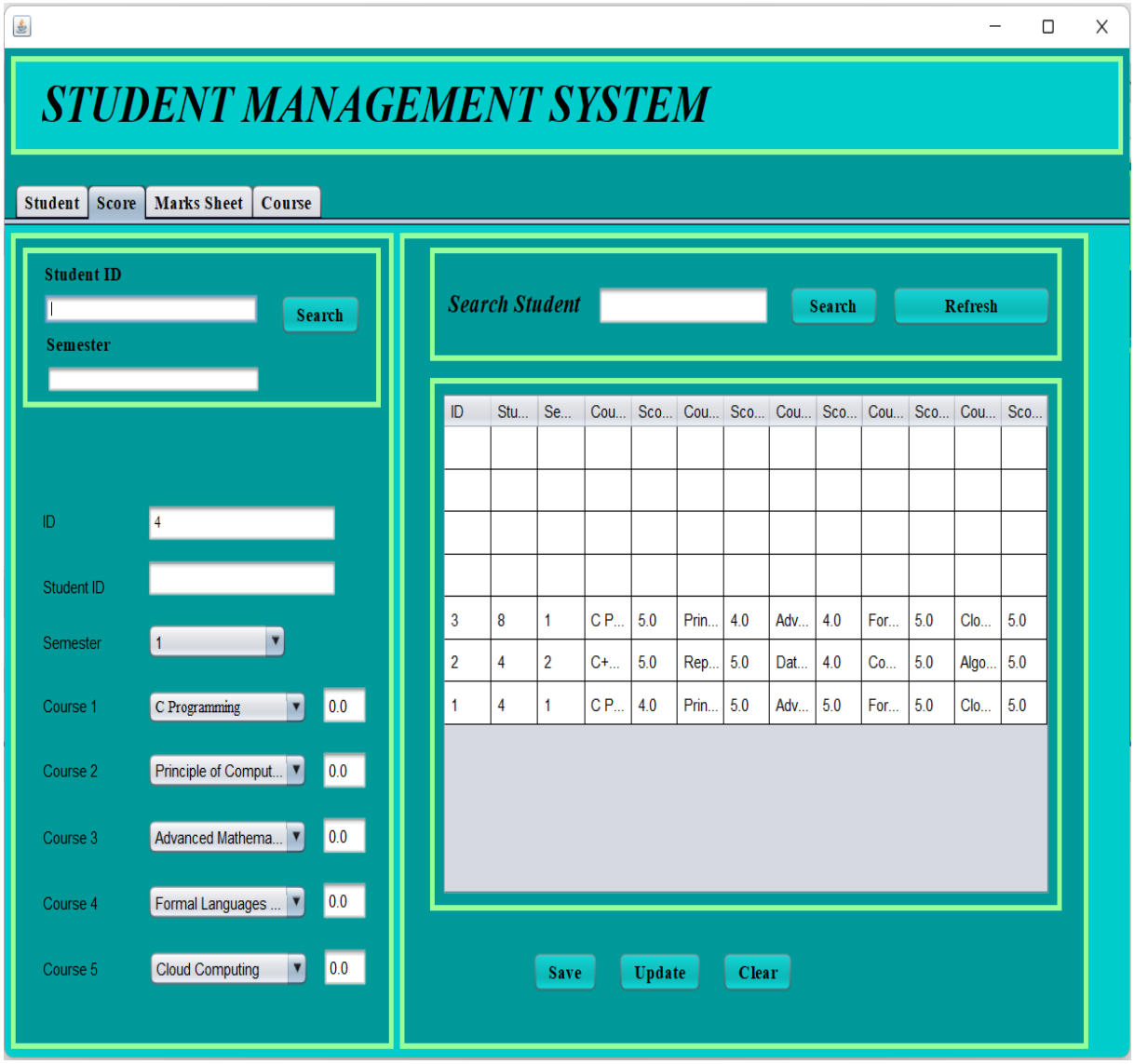
Finally the user can exit the System safely by logging out.

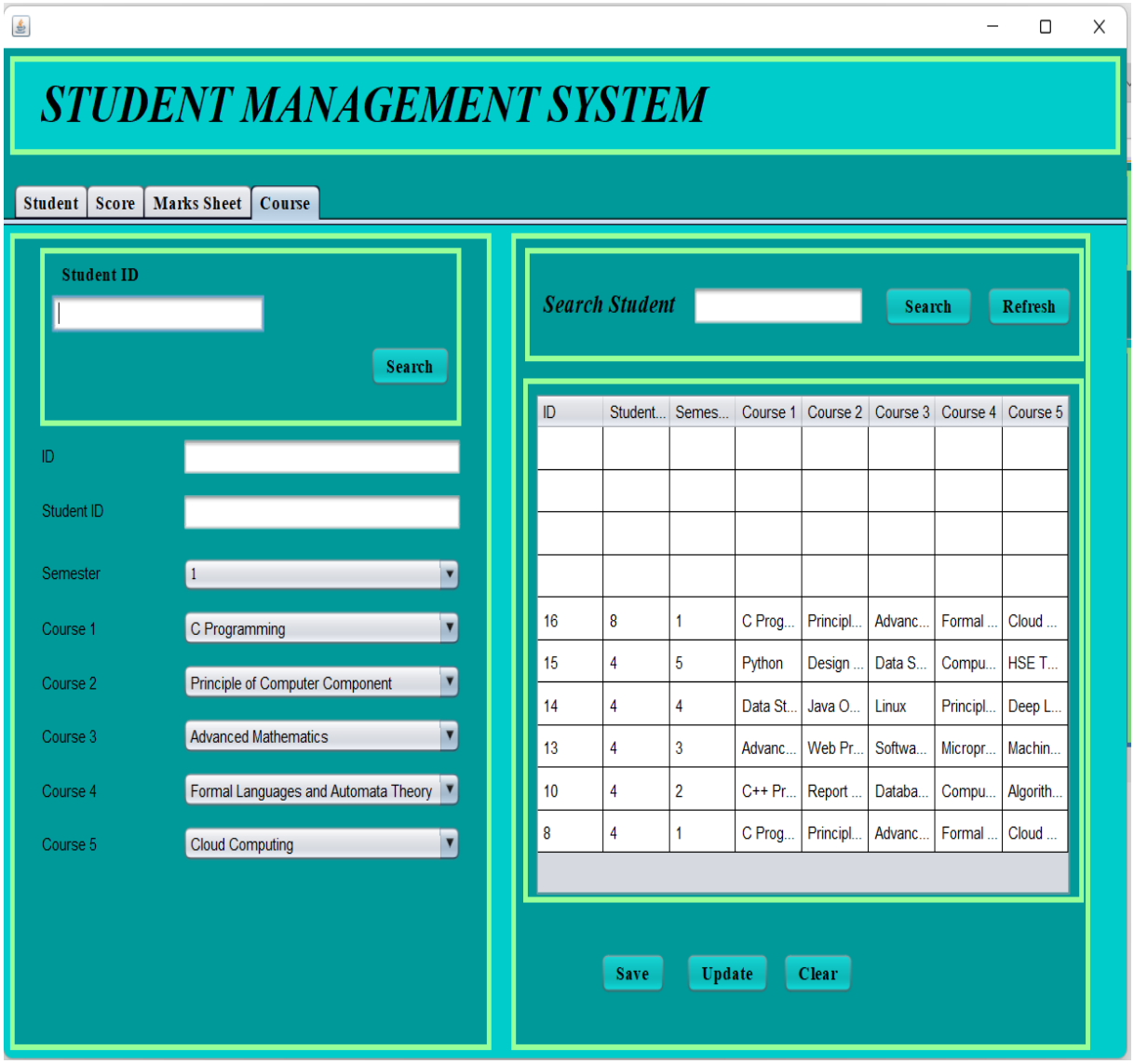
Relational Schema

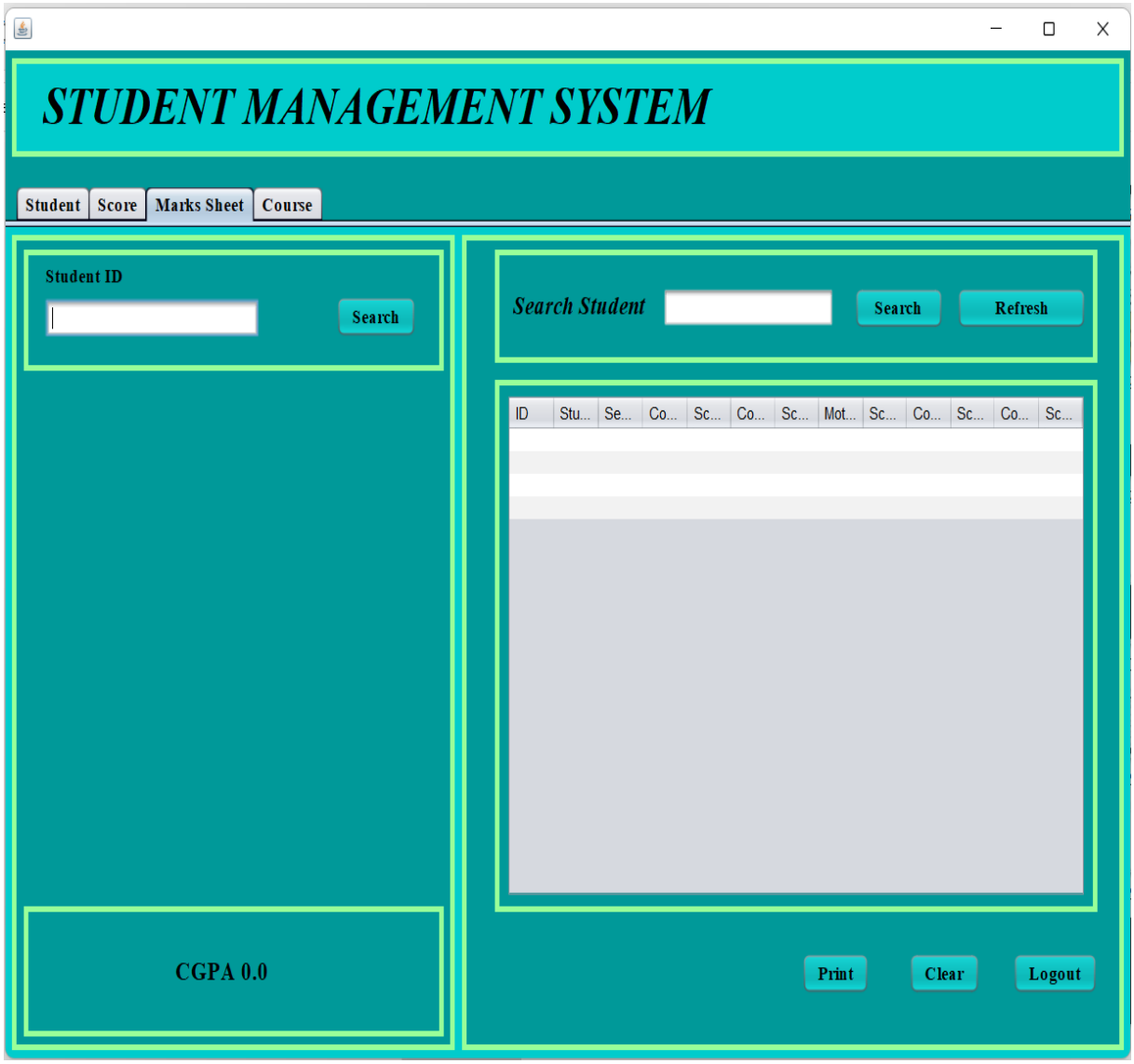


ScreenShots




STUDENT MANAGEMENT SYSTEM
Student
Score
Marks Sheet
Course
Student ID
Search
Semester
Search Student
Search
Refresh
ID
Stu...
Se...
Cou...
Sco...
Cou...
Sco...
Cou...
Sco...
Cou...
Sco...
Cou...
Sco...
3 8 1 C P... 5.0 Prin... 4.0 Adv... 4.0 For... 5.0 Clo... 5.0
2 4 2 C+... 5.0 Rep... 5.0 Dat... 4.0 Co... 5.0 Algo... 5.0
1 4 1 C P... 4.0 Prin... 5.0 Adv... 5.0 For... 5.0 Clo... 5.0
ID
4
Student ID
Semester
1
Course 1
C Programming
0.0
Course 2
Principle of Comput...
0.0
Course 3
Advanced Mathema...
0.0
Course 4
Formal Languages ...
0.0
Course 5
Cloud Computing
0.0
Save
Update
Clear


STUDENT MANAGEMENT SYSTEM
Student
Score
Marks Sheet
Course
Student ID
Search
ID
Student ID
Semester
1
Course 1
C Programming
Course 2
Principle of Computer Component
Course 3
Advanced Mathematics
Course 4
Formal Languages and Automata Theory
Course 5
Cloud Computing
Search Student
Search
Refresh
ID
Student...
Semes...
Course 1
Course 2
Course 3
Course 4
Course 5
16 8 1 C Prog... Principl... Advanc... Formal ... Cloud ...
15 4 5 Python Design ... Data S... Compu... HSE T...
14 4 4 Data St... Java O... Linux Principl... Deep L...
13 4 3 Advanc... Web Pr... Softwa... Micropr... Machin...
10 4 2 C++ Pr... Report ... Databa... Compu... Algorith...
8 4 1 C Prog... Principl... Advanc... Formal ... Cloud ...
Save
Update
Clear


STUDENT MANAGEMENT SYSTEM
Student
Score
Marks Sheet
Course
Student ID
Search
Search Student
Search
Refresh
ID
Stu...
Se...
Co...
Sc...
Co...
Sc...
Mot...
Sc...
Co...
Sc...
Co...
Sc...
CGPA 0.0
Print
Clear
Logout

References

[1] Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems*. Prentice Hall International, 6 edition, 2010.

[2] Raghu Ramakrishnan, Johannes Gehrke, and Johannes Gehrke. *Database management systems*, volume 3. McGraw-Hill New York, 2003.

[3] Kevin Loney. *Oracle database 10g: the complete reference*. McGraw-Hill/Osborne London, 2004.